# Hybrid Continuous Mixtures of Probabilistic Circuits

Dewi Batista
July 18, 2025

# Outline

Part 1: Probabilistic models

Part 2: Probabilistic circuits (PCs)

Part 3: Continuous mixtures of PCs (CMPCs)

Part 4: Hybrid CMPCs

# Part 1: Probabilistic modelling

## Notation

Probability density/mass functions are denoted by $p$; if parameterised then $p_\Theta$.

# Part 1: Probabilistic modelling

## Notation

Probability density/mass functions are denoted by $p$; if parameterised then $p_\Theta$.

## Terminology

Probability density/mass functions are referred to as probability functions.

# Part 1: Probabilistic modelling

## Notation

Probability density/mass functions are denoted by $p$; if parameterised then $p_\Theta$.

## Terminology

Probability density/mass functions are referred to as probability functions.

## More notation

The sample space of a set of random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ is denoted $\Omega_\mathbf{X}$.

We often write $\mathbf{x}$ to denote a realisation $(x_1, \ldots, x_n) \in \Omega_\mathbf{X}$.

# Part 1: Probabilistic modelling

## Definition

A generative probabilistic model $(\mathbf{X}, p_\Theta)$ consists of a finite set $\mathbf{X} = \{X_1, \ldots, X_n\}$ of random variables and a representation

$$p_\Theta : \Omega_{\mathbf{X}} \to \mathbb{R}_{\geq 0}$$

of a probability function over $\mathbf{X}$ parameterised by a parameter tuple $\Theta$.

# Part 1: Probabilistic modelling

## Definition

A generative probabilistic model $(\mathbf{X}, p_\Theta)$ consists of a finite set $\mathbf{X} = \{X_1, \dots, X_n\}$ of random variables and a representation

$$p_\Theta : \Omega_{\mathbf{X}} \to \mathbb{R}_{\geq 0}$$

of a probability function over $\mathbf{X}$ parameterised by a parameter tuple $\Theta$.

Probabilistic queries:

- Evidence (**evi**): $p_\Theta(\mathbf{x})$ where $\mathbf{x} \in \Omega_{\mathbf{X}}$
- Marginal (**marg**): $p_\Theta(\mathbf{x}')$ where $\mathbf{x}' \in \Omega_{\mathbf{X}'}$ with $\mathbf{X}' \subset \mathbf{X}$
- Sampling (**samp**): a stochastic procedure that yields some $\mathbf{x} \in \Omega_{\mathbf{X}}$

# Part 1: Probabilistic modelling

## Definition

A generative probabilistic model $(\mathbf{X}, p_\Theta)$ consists of a finite set $\mathbf{X} = \{X_1, \ldots, X_n\}$ of random variables and a representation

$$p_\Theta : \Omega_\mathbf{X} \to \mathbb{R}_{\geq 0}$$

of a probability function over $\mathbf{X}$ parameterised by a parameter tuple $\Theta$.

Probabilistic queries:

- Evidence (**evi**): $p_\Theta(\mathbf{x})$ where $\mathbf{x} \in \Omega_\mathbf{X}$
- Marginal (**marg**): $p_\Theta(\mathbf{x}')$ where $\mathbf{x}' \in \Omega_{\mathbf{X}'}$ with $\mathbf{X}' \subset \mathbf{X}$
- Sampling (**samp**): a stochastic procedure that yields some $\mathbf{x} \in \Omega_\mathbf{X}$

# Part 1: Probabilistic modelling

## Definition

A generative probabilistic model $(\mathbf{X}, p_\Theta)$ consists of a finite set $\mathbf{X} = \{X_1, \ldots, X_n\}$ of random variables and a representation

$$p_\Theta : \Omega_\mathbf{X} \to \mathbb{R}_{\geq 0}$$

of a probability function over $\mathbf{X}$ parameterised by a parameter tuple $\Theta$.

Probabilistic queries:

- Evidence (**evi**): $p_\Theta(\mathbf{x})$ where $\mathbf{x} \in \Omega_\mathbf{X}$
- Marginal (**marg**): $p_\Theta(\mathbf{x}')$ where $\mathbf{x}' \in \Omega_{\mathbf{X}'}$ with $\mathbf{X}' \subset \mathbf{X}$
- Sampling (**samp**): a stochastic procedure that yields some $\mathbf{x} \in \Omega_\mathbf{X}$

# Part 1: Probabilistic modelling

## Definition

A generative probabilistic model $(\mathbf{X}, p_\Theta)$ consists of a finite set $\mathbf{X} = \{X_1, \ldots, X_n\}$ of random variables and a representation

$$p_\Theta : \Omega_\mathbf{X} \to \mathbb{R}_{\geq 0}$$

of a probability function over $\mathbf{X}$ parameterised by a parameter tuple $\Theta$.

Probabilistic queries:

- Evidence (**evi**): $p_\Theta(\mathbf{x})$ where $\mathbf{x} \in \Omega_\mathbf{X}$
- Marginal (**marg**): $p_\Theta(\mathbf{x}')$ where $\mathbf{x}' \in \Omega_{\mathbf{X}'}$ with $\mathbf{X}' \subset \mathbf{X}$
- Sampling (**samp**): a stochastic procedure that yields some $\mathbf{x} \in \Omega_\mathbf{X}$

# Part 1: Probabilistic modelling

## Definition

A generative probabilistic model $(\mathbf{X}, p_\Theta)$ consists of a finite set $\mathbf{X} = \{X_1, \ldots, X_n\}$ of random variables and a representation

$$p_\Theta : \Omega_\mathbf{X} \to \mathbb{R}_{\geq 0}$$

of a probability function over $\mathbf{X}$ parameterised by a parameter tuple $\Theta$.

Probabilistic queries:

- Evidence (**evi**): $p_\Theta(\mathbf{x})$ where $\mathbf{x} \in \Omega_\mathbf{X}$
- Marginal (**marg**): $p_\Theta(\mathbf{x}')$ where $\mathbf{x}' \in \Omega_{\mathbf{X}'}$ with $\mathbf{X}' \subset \mathbf{X}$
- Sampling (**samp**): a stochastic procedure that yields some $\mathbf{x} \in \Omega_\mathbf{X}$

## Terminology

Probabilistic queries are **answered**.

# Part 1: Probabilistic modelling (example)

## Gaussian mixtures

A univariate Gaussian mixture $(\mathbf{X}, p_{(K,\mu,\sigma,\mathbf{w})})$ is a probabilistic model in which $\mathbf{X} = \{X\}$, so $\Omega_{\mathbf{X}} = \mathbb{R}$, $\mathbf{x} = x$ and

$$p_\Theta(\mathbf{x}) = \sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{x}|\mu_k, \sigma_k^2)$$

where $\Theta = (K, \mu, \sigma, \mathbf{w})$ consists of the number of components $K \in \mathbb{N}$, means $\mu = (\mu_1, \ldots, \mu_K) \in \mathbb{R}^K$, standard deviations $\sigma = (\sigma_1, \ldots, \sigma_K) \in \mathbb{R}^K_{>0}$ and weights $\mathbf{w} = (w_1, \ldots, w_K) \in \mathbb{R}^K_{\geq 0}$ with $\sum_{k=1}^K w_k = 1$.

# Part 1: Probabilistic modelling (example)

## Gaussian mixtures

A univariate Gaussian mixture $(\mathbf{X}, p_{(K,\mu,\sigma,\mathbf{w})})$ is a probabilistic model in which $\mathbf{X} = \{X\}$, so $\Omega_{\mathbf{X}} = \mathbb{R}$, $\mathbf{x} = x$ and

$$p_{\Theta}(\mathbf{x}) = \sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{x}|\mu_k, \sigma_k^2)$$

# Part 1: Probabilistic modelling (example)

## Gaussian mixtures

A univariate Gaussian mixture $(\mathbf{X}, p_{(K,\mu,\sigma,\mathbf{w})})$ is a probabilistic model in which $\mathbf{X} = \{X\}$, so $\Omega_{\mathbf{X}} = \mathbb{R}$, $\mathbf{x} = x$ and

$$p_{\Theta}(\mathbf{x}) = \sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{x}|\mu_k, \sigma_k^2)$$

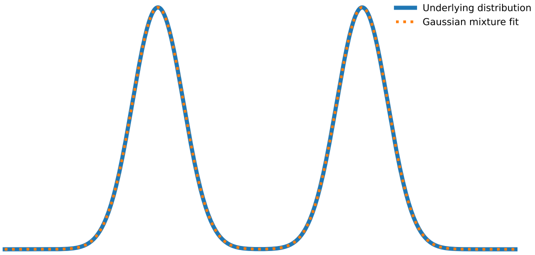



Figure: A Gaussian mixture with $K = 2$ fit to a bimodal distribution.

# Part 1: Probabilistic modelling (example)

## Gaussian mixtures

A univariate Gaussian mixture $(\mathbf{X}, p_{(K,\mu,\sigma,\mathbf{w})})$ is a probabilistic model in which $\mathbf{X} = \{X\}$, so $\Omega_{\mathbf{X}} = \mathbb{R}$, $\mathbf{x} = x$ and

$$p_{\Theta}(\mathbf{x}) = \sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{x}|\mu_k, \sigma_k^2)$$

## The class of univariate Gaussian mixtures

We often talk about classes of probabilistic models. For example,

$$\mathsf{GM}(\mathbf{X}) = \left\{ p_{(K,\mu,\sigma,\mathbf{w})} : K \in \mathbb{N}, \mu \in \mathbb{R}^K, \sigma, \mathbf{w} \in \mathbb{R}^K_{>0} \text{ and } \sum_{k=1}^{K} w_k = 1 \right\}$$

# Part 1: Probabilistic modelling (trifecta of goodness)

The trifecta of goodness for a class of probabilistic models:

1. **Tractability**: how efficiently the model class can answer queries
2. **Expressivity**: how precisely a model class can fit distributions
3. **Exp.-efficiency**: how efficiently a model class can fit distributions

# Part 1: Probabilistic modelling (trifecta of goodness)

The trifecta of goodness for a class of probabilistic models:

1. **Tractability**: how efficiently the model class can answer queries
2. **Expressivity**: how precisely a model class can fit distributions
3. **Exp.-efficiency**: how efficiently a model class can fit distributions

# Part 1: Probabilistic modelling (trifecta of goodness)

The trifecta of goodness for a class of probabilistic models:

1. **Tractability**: how efficiently the model class can answer queries
2. **Expressivity**: how precisely a model class can fit distributions
3. **Exp.-efficiency**: how efficiently a model class can fit distributions

# Part 1: Probabilistic modelling (trifecta of goodness)

The trifecta of goodness for a class of probabilistic models:

1. **Tractability**: how efficiently the model class can answer queries
2. **Expressivity**: how precisely a model class can fit distributions
3. **Exp.-efficiency**: how efficiently a model class can fit distributions

# Part 1: Probabilistic modelling (trifecta of goodness)

The trifecta of goodness for a class of probabilistic models:

1. **Tractability**: how efficiently the model class can answer queries
2. **Expressivity**: how precisely a model class can fit distributions
3. **Exp.-efficiency**: how efficiently a model class can fit distributions

| **Model\Probabilistic query** | **samp** | **evi** | **marg** |
|---|---|---|---|
| Bayesian networks [5] | ✓ | ✓ | ✗ |
| Gaussian mixtures | ✓ | ✓ | ✓ |
| Probabilistic circuits | ✓ | ✓ | ✓ |

Table: The tractability of classes of probabilistic models.

# Part 1: Probabilistic modelling (trifecta of goodness)

The trifecta of goodness for a class of probabilistic models:

1. **Tractability**: how efficiently the model class can answer queries
2. **Expressivity**: how precisely a model class can fit distributions
3. **Exp.-efficiency**: how efficiently a model class can fit distributions

# Part 1: Probabilistic modelling (trifecta of goodness)

The trifecta of goodness for a class of probabilistic models:

1. **Tractability**: how efficiently the model class can answer queries
2. **Expressivity**: how precisely a model class can fit distributions
3. **Exp.-efficiency**: how efficiently a model class can fit distributions

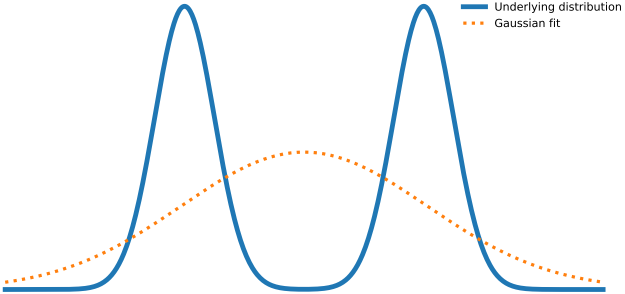



Figure: A Gaussian fit to a bimodal distribution.

# Part 1: Probabilistic modelling (trifecta of goodness)

The trifecta of goodness for a class of probabilistic models:

1. **Tractability**: how efficiently the model class can answer queries
2. **Expressivity**: how precisely a model class can fit distributions
3. **Exp.-efficiency**: how efficiently a model class can fit distributions

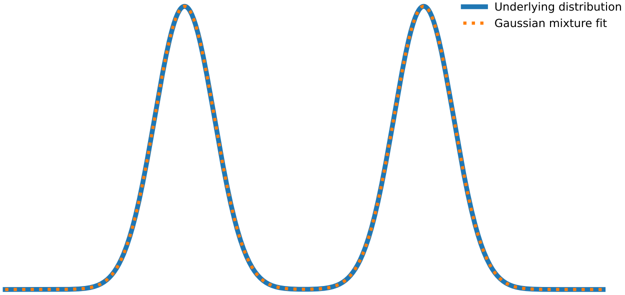



Figure: A Gaussian mixture fit to a bimodal distribution.

# Part 1: Probabilistic modelling (trifecta of goodness)

The trifecta of goodness for a class of probabilistic models:

1. **Tractability**: how efficiently the model class can answer queries
2. **Expressivity**: how precisely a model class can fit distributions
3. **Exp.-efficiency**: how efficiently a model class can fit distributions

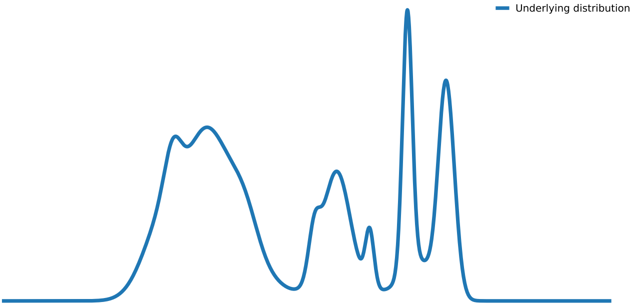



Figure: A complex distribution.

# Part 1: Probabilistic modelling (trifecta of goodness)

The trifecta of goodness for a class of probabilistic models:

1. **Tractability**: how efficiently the model class can answer queries
2. **Expressivity**: how precisely a model class can fit distributions
3. **Exp.-efficiency**: how efficiently a model class can fit distributions

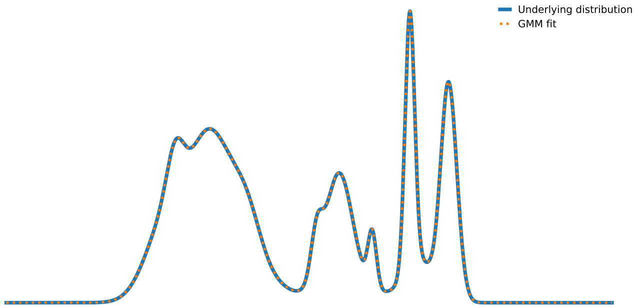



Figure: A 20-component Gaussian mixture fit to a complex distribution.

# Part 1: Probabilistic modelling (trifecta of goodness)

The trifecta of goodness for a class of probabilistic models:

1. **Tractability**: how efficiently the model class can answer queries
2. **Expressivity**: how precisely a model class can fit distributions
3. **Exp.-efficiency**: how efficiently a model class can fit distributions

**Universal distribution approximation**

"A Gaussian mixture model is a universal approximator of densities, in the sense that any smooth density can be approximated with any specific non-zero amount of error by a Gaussian mixture model with enough components." [3, Page 67]

The class of Gaussian mixtures has perfect expressivity!

# Part 1: Probabilistic modelling (trifecta of goodness)

The trifecta of goodness for a class of probabilistic models:

1. **Tractability**: how efficiently the model class can answer queries
2. **Expressivity**: how precisely a model class can fit distributions
3. **Exp.-efficiency**: how efficiently a model class can fit distributions

## Universal distribution approximation

"A Gaussian mixture model is a universal approximator of densities, in the sense that any smooth density can be approximated with any specific non-zero amount of error by a Gaussian mixture model with enough components." [3, Page 67]

The class of Gaussian mixtures has perfect expressivity!

# Part 1: Probabilistic modelling (trifecta of goodness)

The trifecta of goodness for a class of probabilistic models:

1. **Tractability**: how efficiently the model class can answer queries
2. **Expressivity**: how precisely a model class can fit distributions
3. **Exp.-efficiency**: how efficiently a model class can fit distributions

# Part 1: Probabilistic modelling (trifecta of goodness)

The trifecta of goodness for a class of probabilistic models:

1. **Tractability**: how efficiently the model class can answer queries
2. **Expressivity**: how precisely a model class can fit distributions
3. **Exp.-efficiency**: how efficiently a model class can fit distributions

**Example:** The number of components in a Gaussian mixture needed to fit complex distributions is high in practice, i.e. low expressive-efficiency!

# Part 1: Probabilistic modelling (trifecta of goodness)

The trifecta of goodness for a class of probabilistic models:

1. **Tractability**: how efficiently the model class can answer queries
2. **Expressivity**: how precisely a model class can fit distributions
3. **Exp.-efficiency**: how efficiently a model class can fit distributions

**Example:** The number of components in a Gaussian mixture needed to fit complex distributions is high in practice, i.e. low expressive-efficiency!

Is there a class of probabilistic models which offers a bit of all three?

# Part 1: Probabilistic modelling (trifecta of goodness)

The trifecta of goodness for a class of probabilistic models:

1. **Tractability**: how efficiently the model class can answer queries
2. **Expressivity**: how precisely a model class can fit distributions
3. **Exp.-efficiency**: how efficiently a model class can fit distributions

**Example:** The number of components in a Gaussian mixture needed to fit complex distributions is high in practice, i.e. low expressive-efficiency!

Is there a class of probabilistic models which offers a bit of all three?

Probabilistic circuits (PCs)!

# Part 2: Probabilistic circuits (PCs)

## Beginning of overview

A probabilistic circuit $(\mathbf{X}, p_{(\mathcal{G},\mathbf{w},\theta)})$ is a generative probabilistic model whose parameter tuple $\Theta = (\mathcal{G}, \mathbf{w}, \theta)$ consists of a weighted graph $\mathcal{G}$, tuple of weights $\mathbf{w}$ and a parameter tuple $\theta$ pertaining to the probability functions of its input nodes.

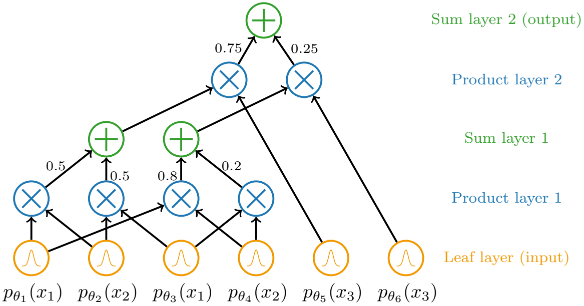



Figure: A PC over $\mathbf{X} = \{X_1, X_2, X_3\}$ with six input nodes.

Input nodes in orange, product nodes in blue and sum nodes in green.

# Part 2: Probabilistic circuits (PCs)

## Beginning of overview

A probabilistic circuit $(\mathbf{X}, p_{(\mathcal{G},\mathbf{w},\theta)})$ is a generative probabilistic model whose parameter tuple $\Theta = (\mathcal{G}, \mathbf{w}, \theta)$ consists of a weighted graph $\mathcal{G}$, tuple of weights $\mathbf{w}$ and a parameter tuple $\theta$ pertaining to the probability functions of its input nodes.

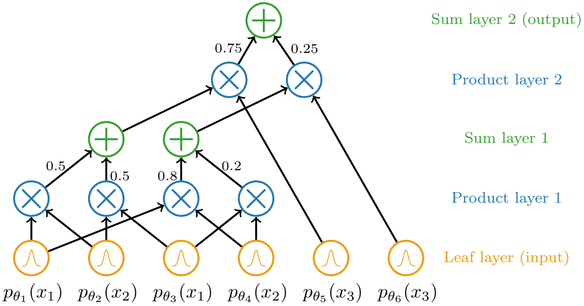



Figure: A PC over $\mathbf{X} = \{X_1, X_2, X_3\}$ with six input nodes.

Input nodes in orange, product nodes in blue and sum nodes in green.

## Comment on weights

The weights of the edges pointing to a given sum node must sum to 1.

# Part 2: Probabilistic circuits (PCs)

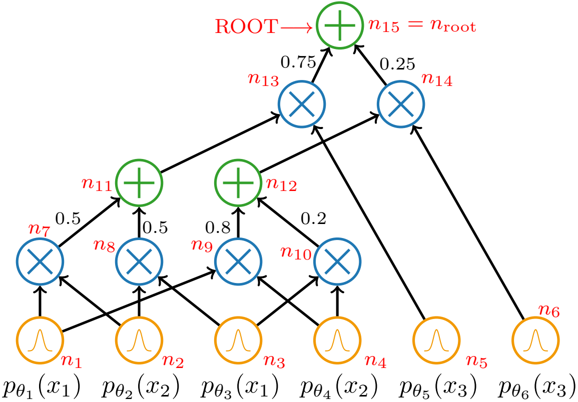



Figure: A PC over $\mathbf{X} = \{X_1, X_2, X_3\}$ with six input nodes.

Input nodes in orange, product nodes in blue and sum nodes in green.

# Part 2: Probabilistic circuits (PCs)

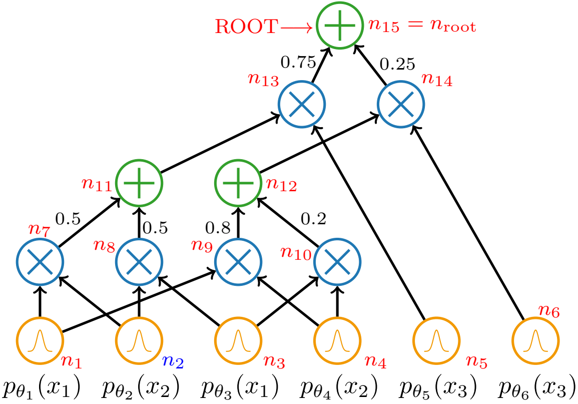



Figure: A PC over $\mathbf{X} = \{X_1, X_2, X_3\}$ with six input nodes.

Input nodes in orange, product nodes in blue and sum nodes in green.

Associated function of node $n_2$:

$$p_{n_2}(x_2) = p_{\theta_2}(x_2)$$

# Part 2: Probabilistic circuits (PCs)

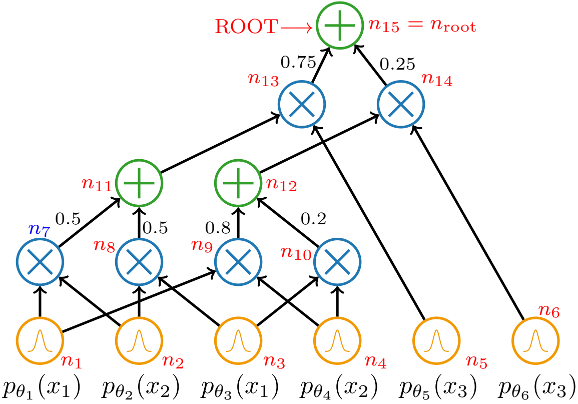



Figure: A PC over $\mathbf{X} = \{X_1, X_2, X_3\}$ with six input nodes.

Input nodes in orange, product nodes in blue and sum nodes in green.

Associated function of node $n_7$:

$$p_{n_7}(x_1, x_2) = p_{\theta_1}(x_1)p_{\theta_2}(x_2)$$

# Part 2: Probabilistic circuits (PCs)

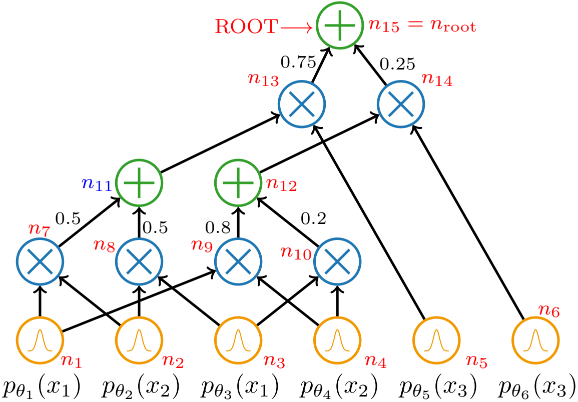



Figure: A PC over $\mathbf{X} = \{X_1, X_2, X_3\}$ with six input nodes.

Input nodes in orange, product nodes in blue and sum nodes in green.

Associated function of node $n_{11}$:

$$
\begin{aligned}
p_{n_{11}}(x_1, x_2) &= 0.5 p_{n_7}(x_1, x_2) + 0.5 p_{n_8}(x_1, x_2) \\
&= 0.5 p_{\theta_1}(x_1) p_{\theta_2}(x_2) + 0.5 p_{\theta_3}(x_1) p_{\theta_2}(x_2)
\end{aligned}
$$

# Part 2: Probabilistic circuits (PCs)

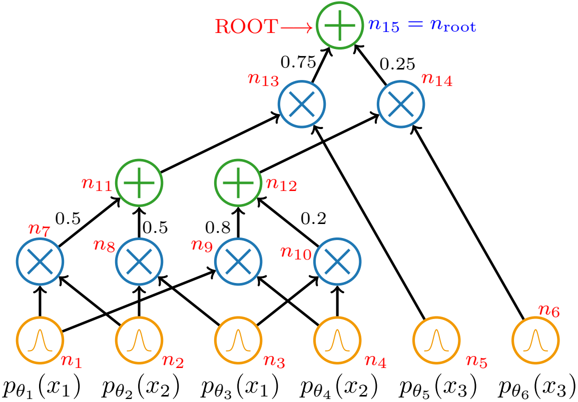



Figure: A PC over $\mathbf{X} = \{X_1, X_2, X_3\}$ with six input nodes.

Input nodes in orange, product nodes in blue and sum nodes in green.

Associated function of node $n_{15}$ (the root node):

$$
\begin{aligned}
p_{n_{15}}(x_1, x_2, x_3) = {} & \frac{15}{40} p_{\theta_1}(x_1) p_{\theta_2}(x_2) p_{\theta_5}(x_3) + \frac{15}{40} p_{\theta_3}(x_1) p_{\theta_2}(x_2) p_{\theta_5}(x_3) \\
& + \frac{8}{40} p_{\theta_1}(x_1) p_{\theta_4}(x_2) p_{\theta_6}(x_3) + \frac{2}{40} p_{\theta_3}(x_1) p_{\theta_4}(x_2) p_{\theta_6}(x_3)
\end{aligned}
$$

# Part 2: Probabilistic circuits (PCs)

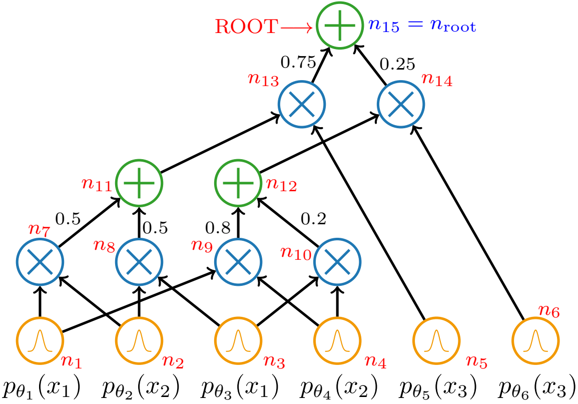



Figure: A PC over $\mathbf{X} = \{X_1, X_2, X_3\}$ with six input nodes.

Input nodes in orange, product nodes in blue and sum nodes in green.

What is $p_{(\mathcal{G},\mathbf{w},\theta)}$ for a PC?

# Part 2: Probabilistic circuits (PCs)

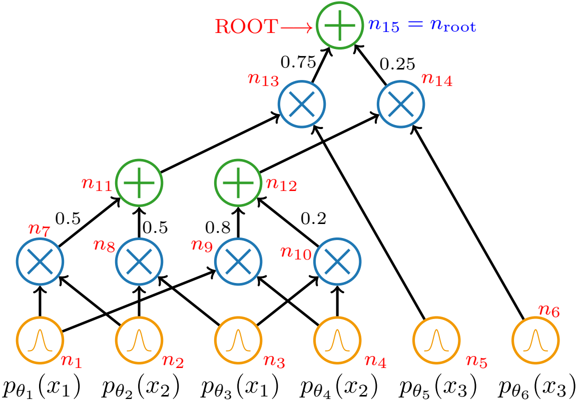



Figure: A PC over $\mathbf{X} = \{X_1, X_2, X_3\}$ with six input nodes.

Input nodes in orange, product nodes in blue and sum nodes in green.

What is $p_{(\mathcal{G},\mathbf{w},\theta)}$ for a PC? The associated function of the root node, $n_{\text{root}}$!

# Part 2: Probabilistic circuits (PCs)

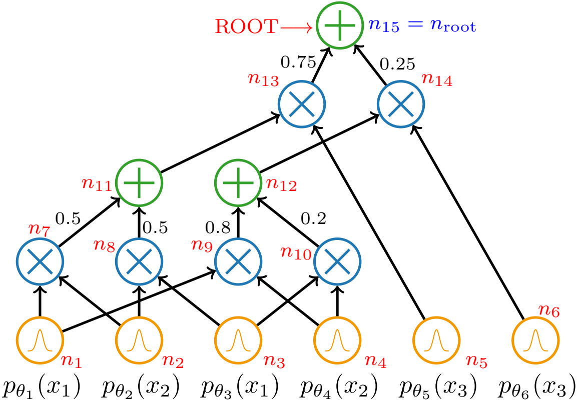


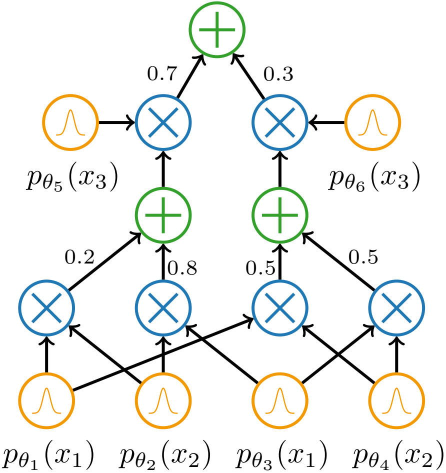

Figure: A PC over $\mathbf{X} = \{X_1, X_2, X_3\}$ with six input nodes.

Input nodes in orange, product nodes in blue and sum nodes in green.

What is $p_{(\mathcal{G},\mathbf{w},\theta)}$ for a PC? The associated function of the root node, $n_{\mathrm{root}}$!

$$p_{n_{15}}(x_1, x_2, x_3) = \frac{15}{40} p_{\theta_1}(x_1) p_{\theta_2}(x_2) p_{\theta_5}(x_3) + \frac{15}{40} p_{\theta_3}(x_1) p_{\theta_2}(x_2) p_{\theta_5}(x_3) + \frac{8}{40} p_{\theta_1}(x_1) p_{\theta_4}(x_2) p_{\theta_6}(x_3) + \frac{2}{40} p_{\theta_3}(x_1) p_{\theta_4}(x_2) p_{\theta_6}(x_3)$$

# Part 2: Probabilistic circuits (answering **evi** queries)




**Computational graph $\mathcal{G}$:**
Displayed to the left

**Weights w:**
Displayed to the left

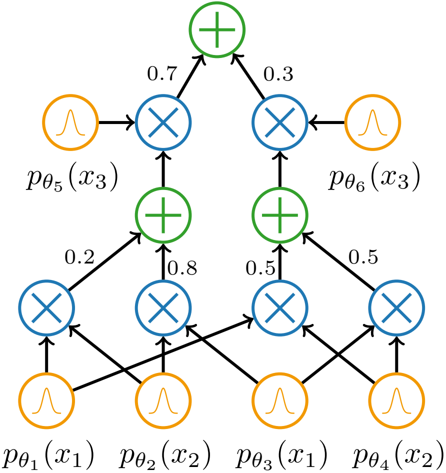
**Input nodes:**
$p_{\theta_i}(x) = \text{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_2, x_3) = (0, 1, 1)$

**Evidence query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1, 1) =$ **?**

# Part 2: Probabilistic circuits (answering **evi** queries)




**Computational graph $\mathcal{G}$:**
Displayed to the left

**Weights $\mathbf{w}$:**
Displayed to the left

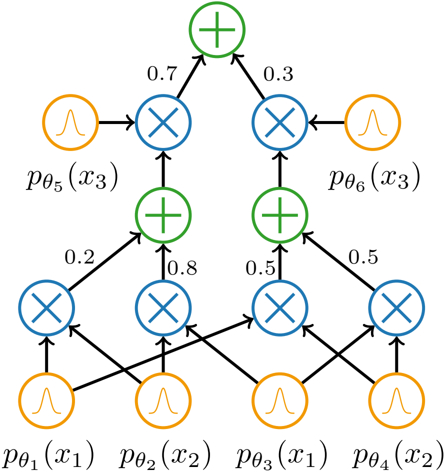
**Input nodes:**
$p_{\theta_i}(x) = \text{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_2, x_3) = (0, 1, 1)$

**Evidence query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1, 1) =$ **?**

# Part 2: Probabilistic circuits (answering **evi** queries)




**Computational graph $\mathcal{G}$:**
Displayed to the left

**Weights $\mathrm{w}$:**
Displayed to the left

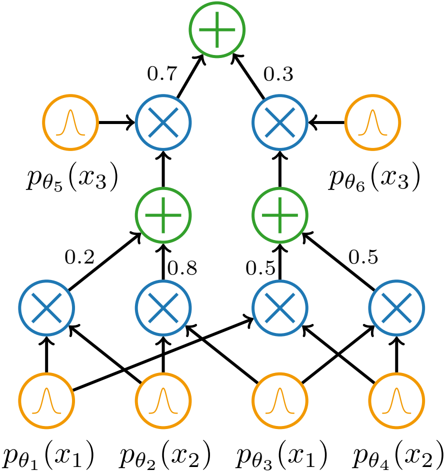
**Input nodes:**
$p_{\theta_i}(x) = \mathsf{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_2, x_3) = (0, 1, 1)$

**Evidence query:**
$p_{(\mathcal{G},\mathrm{w},\theta)}(0, 1, 1) = \textbf{?}$

# Part 2: Probabilistic circuits (answering **evi** queries)




**Computational graph $\mathcal{G}$:**
Displayed to the left

**Weights $\mathbf{w}$:**
Displayed to the left

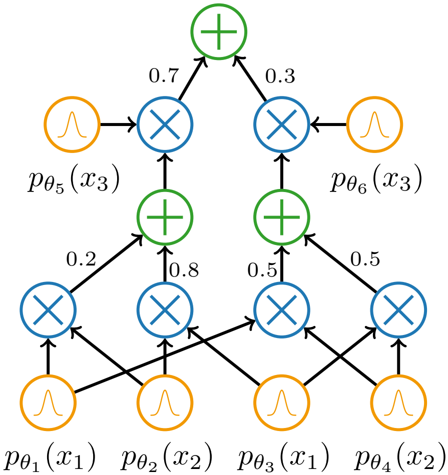
**Input nodes:**
$p_{\theta_i}(x) = \text{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_2, x_3) = (0, 1, 1)$

**Evidence query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1, 1) = \mathbf{?}$

# Part 2: Probabilistic circuits (answering **evi** queries)




**Computational graph $\mathcal{G}$:**
Displayed to the left

**Weights $\mathbf{w}$:**
Displayed to the left

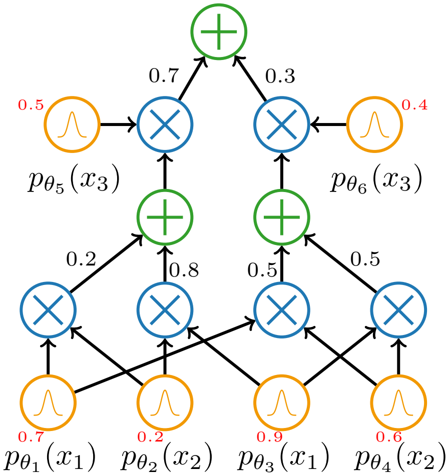
**Input nodes:**
$p_{\theta_i}(x) = \text{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_2, x_3) = (0, 1, 1)$

**Evidence query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1, 1) =$ **?**

# Part 2: Probabilistic circuits (answering **evi** queries)




**Computational graph $\mathcal{G}$:**
Displayed to the left

**Weights $\mathbf{w}$:**
Displayed to the left

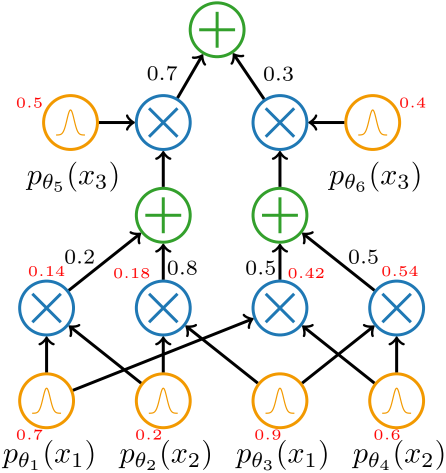
**Input nodes:**
$p_{\theta_i}(x) = \mathsf{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_2, x_3) = (0, 1, 1)$

**Evidence query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1, 1) = \textbf{?}$

# Part 2: Probabilistic circuits (answering **evi** queries)




**Computational graph $\mathcal{G}$:**
Displayed to the left

**Weights $\mathbf{w}$:**
Displayed to the left

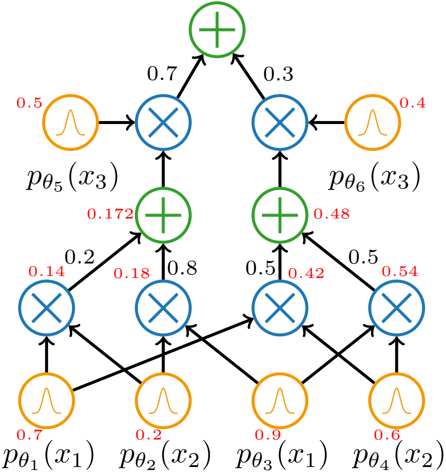
**Input nodes:**
$p_{\theta_i}(x) = \text{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_2, x_3) = (0, 1, 1)$

**Evidence query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1, 1) = $ **?**

# Part 2: Probabilistic circuits (answering **evi** queries)




**Computational graph $\mathcal{G}$:**
Displayed to the left

**Weights $\mathbf{w}$:**
Displayed to the left

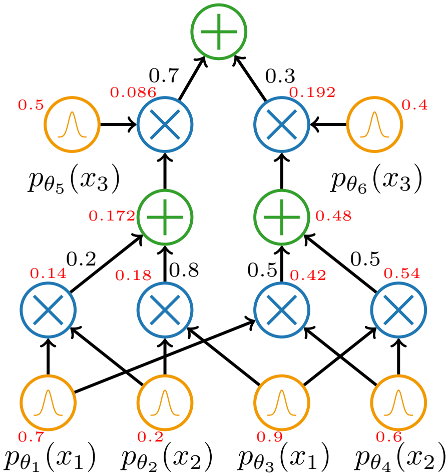
**Input nodes:**
$p_{\theta_i}(x) = \text{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_2, x_3) = (0, 1, 1)$

**Evidence query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1, 1) =$ **?**

# Part 2: Probabilistic circuits (answering **evi** queries)




**Computational graph $\mathcal{G}$:**
Displayed to the left

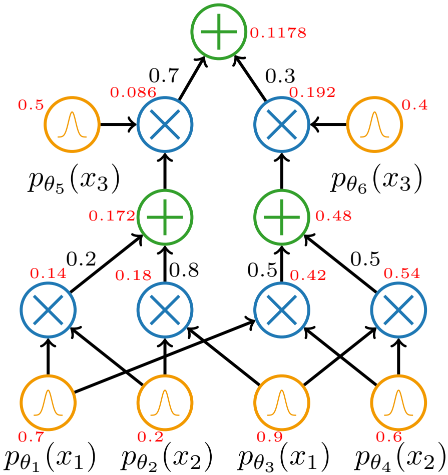
**Weights $\mathbf{w}$:**
Displayed to the left

**Input nodes:**
$p_{\theta_i}(x) = \text{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_2, x_3) = (0, 1, 1)$

**Evidence query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1, 1) =$ **?**




**Computational graph $\mathcal{G}$:**
Displayed to the left

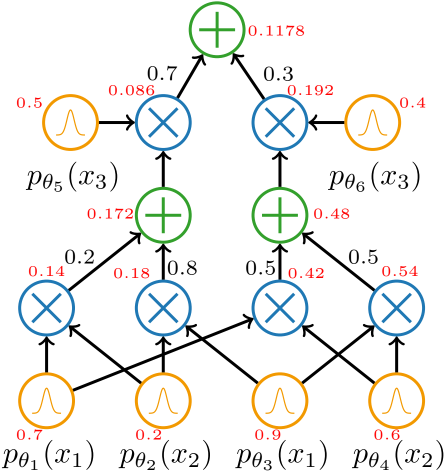
**Weights $\mathbf{w}$:**
Displayed to the left

**Input nodes:**
$p_{\theta_i}(x) = \text{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_2, x_3) = (0, 1, 1)$

**Evidence query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1, 1) =$ **?**




**Computational graph $\mathcal{G}$:**
Displayed to the left

**Weights $\mathbf{w}$:**
Displayed to the left

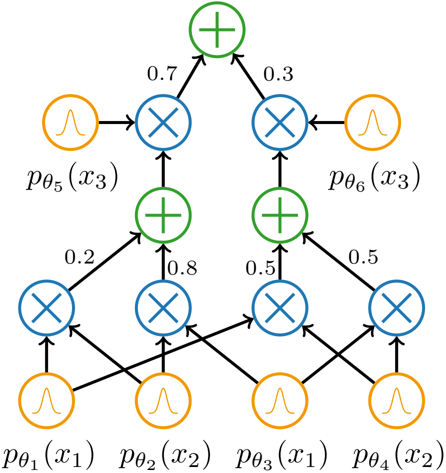
**Input nodes:**
$p_{\theta_i}(x) = \text{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_2, x_3) = (0, 1, 1)$

**Evidence query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1, 1) = 0.1178$

# Part 2: Probabilistic circuits (answering **marg** queries)




**Computational graph $\mathcal{G}$:**
Displayed to the left

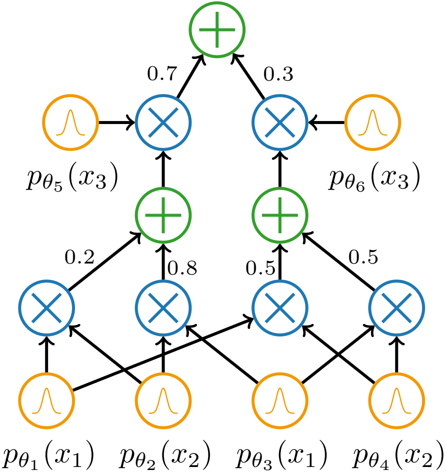
**Weights $\mathbf{w}$:**
Displayed to the left

**Input nodes:**
$p_{\theta_i}(x) = \text{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_3) = (0, 1)$

**Marginal query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1) =$ **?**




**Computational graph $\mathcal{G}$:**
Displayed to the left

**Weights $\mathbf{w}$:**
Displayed to the left

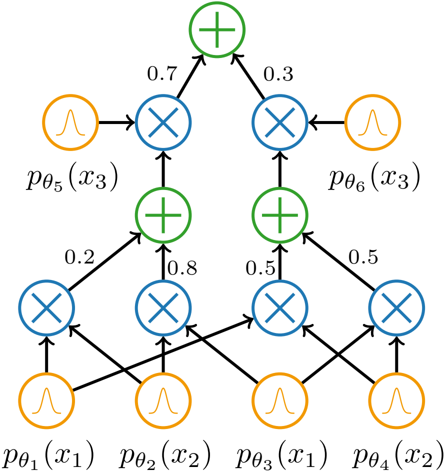
**Input nodes:**
$p_{\theta_i}(x) = \text{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_3) = (0, 1)$

**Marginal query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1) = ?$

# Part 2: Probabilistic circuits (answering **marg** queries)




**Computational graph $\mathcal{G}$:**
Displayed to the left

**Weights $\mathbf{w}$:**
Displayed to the left

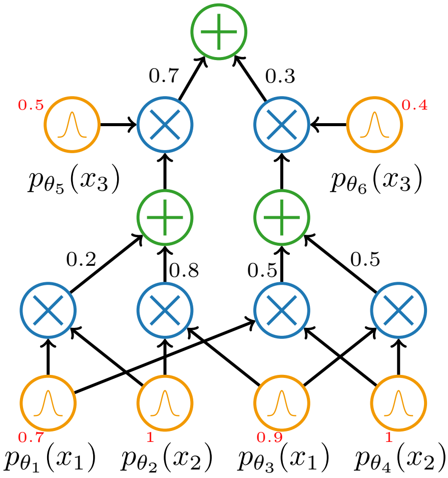
**Input nodes:**
$p_{\theta_i}(x) = \text{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_3) = (0, 1)$

**Marginal query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1) =$ **?**

# Part 2: Probabilistic circuits (answering **marg** queries)




**Computational graph $\mathcal{G}$:**
Displayed to the left

**Weights $\mathbf{w}$:**
Displayed to the left

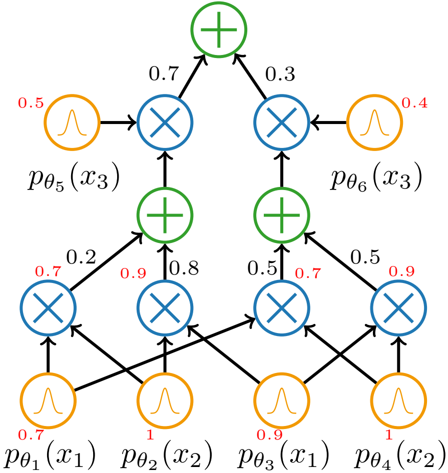
**Input nodes:**
$p_{\theta_i}(x) = \mathsf{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_3) = (0, 1)$

**Marginal query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1) =$ **?**

# Part 2: Probabilistic circuits (answering **marg** queries)




**Computational graph $\mathcal{G}$:**
Displayed to the left

**Weights $\mathbf{w}$:**
Displayed to the left

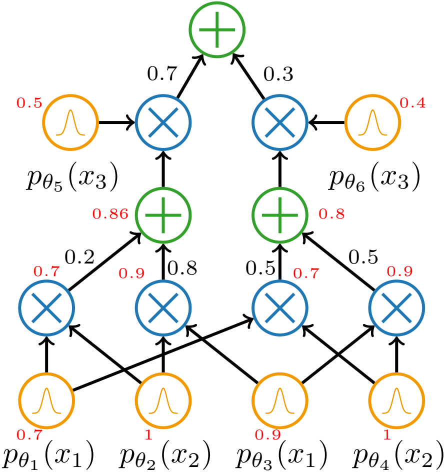
**Input nodes:**
$p_{\theta_i}(x) = \text{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_3) = (0, 1)$

**Marginal query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1) =$ **?**

# Part 2: Probabilistic circuits (answering **marg** queries)




**Computational graph $\mathcal{G}$:**
Displayed to the left

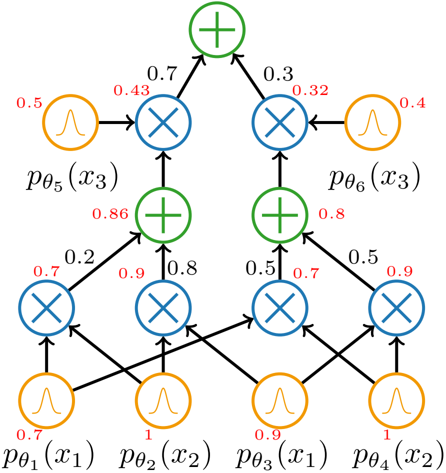
**Weights $\mathbf{w}$:**
Displayed to the left

**Input nodes:**
$p_{\theta_i}(x) = \text{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_3) = (0, 1)$

**Marginal query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1) =$ **?**




**Computational graph $\mathcal{G}$:**
Displayed to the left

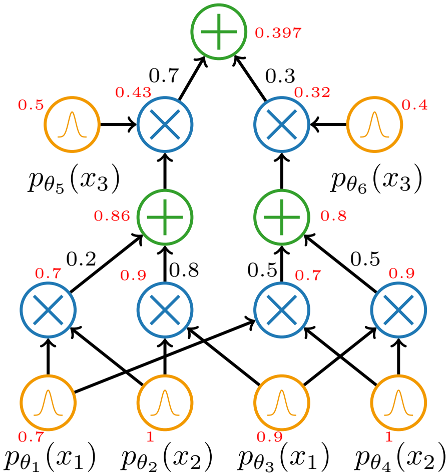
**Weights $\mathbf{w}$:**
Displayed to the left

**Input nodes:**
$p_{\theta_i}(x) = \mathrm{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_3) = (0, 1)$

**Marginal query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1) = \textbf{?}$




**Computational graph $\mathcal{G}$:**
Displayed to the left

**Weights $\mathbf{w}$:**
Displayed to the left

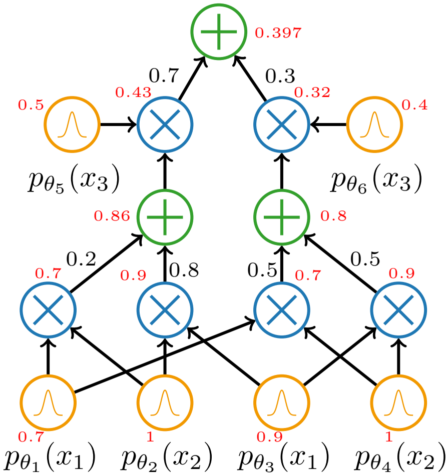
**Input nodes:**
$p_{\theta_i}(x) = \text{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_3) = (0, 1)$

**Marginal query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1) =$ **?**

# Part 2: Probabilistic circuits (answering **marg** queries)




**Computational graph $\mathcal{G}$:**
Displayed to the left

**Weights $\mathbf{w}$:**
Displayed to the left

**Input nodes:**
$p_{\theta_i}(x) = \text{Ber}(x|\theta_i)$ with

$$\theta_1 = 0.3 \quad \theta_2 = 0.2 \quad \theta_3 = 0.1$$
$$\theta_4 = 0.6 \quad \theta_5 = 0.5 \quad \theta_6 = 0.4$$

**Realisation:**
$(x_1, x_3) = (0, 1)$

**Marginal query:**
$p_{(\mathcal{G},\mathbf{w},\theta)}(0, 1) = 0.397$

# Part 2: Probabilistic circuits (overview)

1. **Tractability**: how efficiently the model class can answer queries

**evi** and **marg** queries are linear in the number of edges in the PC!

2. **Expressivity**: how precisely a model class can fit distributions

At least as expressive as Gaussian mixtures which have perfect expressivity!

3. **Expressive-efficiency**: how efficiently a model class can fit distributions

Entirely empirically-motivated

# Part 2: Probabilistic circuits (overview)

1. **Tractability**: how efficiently the model class can answer queries

**evi** and **marg** queries are linear in the number of edges in the PC!

2. **Expressivity**: how precisely a model class can fit distributions

At least as expressive as Gaussian mixtures which have perfect expressivity!

3. **Expressive-efficiency**: how efficiently a model class can fit distributions

Entirely empirically-motivated

# Part 2: Probabilistic circuits (overview)

1. **Tractability**: how efficiently the model class can answer queries

**evi** and **marg** queries are linear in the number of edges in the PC!

2. **Expressivity**: how precisely a model class can fit distributions

At least as expressive as Gaussian mixtures which have perfect expressivity!

3. **Expressive-efficiency**: how efficiently a model class can fit distributions

Entirely empirically-motivated

# Part 2: Probabilistic circuits (overview)

1. **Tractability**: how efficiently the model class can answer queries

**evi** and **marg** queries are linear in the number of edges in the PC!

2. **Expressivity**: how precisely a model class can fit distributions

At least as expressive as Gaussian mixtures which have perfect expressivity!

3. **Expressive-efficiency**: how efficiently a model class can fit distributions

Entirely empirically-motivated

**Learning PCs (so $\mathcal{G}, \mathbf{w}$ and $\theta$) from data**

Tons of methods. Some inspired by learning Bayesian networks from data.

# Part 2: Probabilistic circuits (applications)

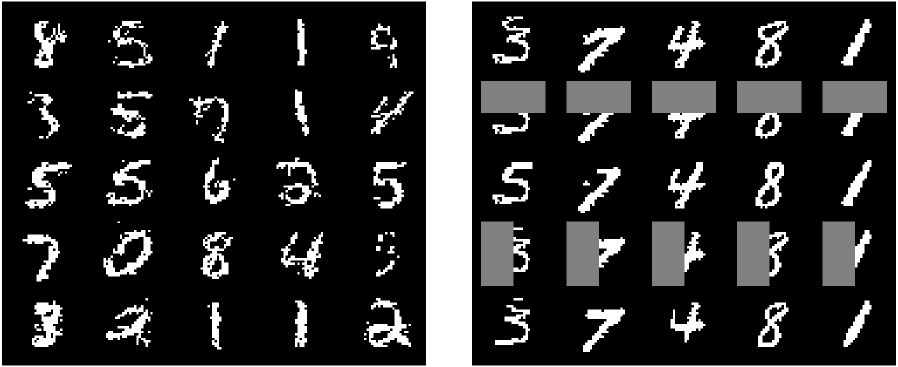
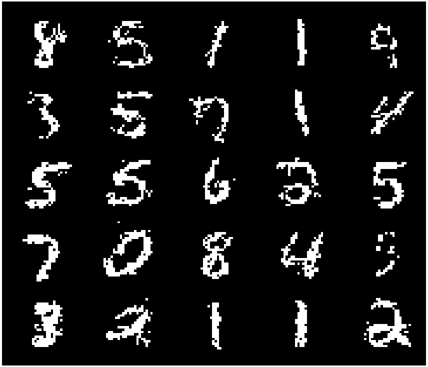


Figure: Sampling (left) and in-painting (right). Taken from [2, Figure 2].

# Part 2: Probabilistic circuits (applications)



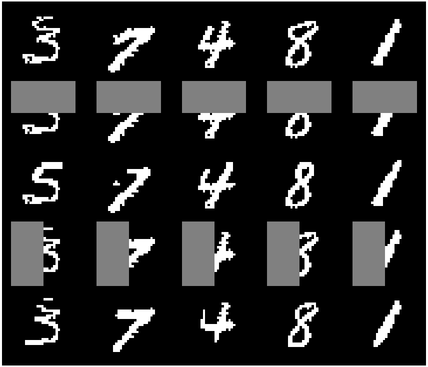


Figure: Sampling (left) and in-painting (right). Taken from [2, Figure 2].

## Classification

Assign label according to $\arg\max_{y \in \Omega_Y} p(y|\mathbf{x}) = \arg\max_{y \in \Omega_Y} p(\mathbf{x}, y)$.

# Part 3: Continuous mixtures of PCs (motivation)

## Definition

A continuous mixture of probabilistic circuits (CMPC) is a PC of the form

$$p(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} p_\phi(\mathbf{x}|\mathbf{z}_i)$$

where $N \in \mathbb{N}$, $\mathbf{z}_1, \ldots, \mathbf{z}_N \sim p(\mathbf{z})$ with $\mathbf{Z} \sim \mathcal{N}(0, I_d)$ for some latent dimension $d \in \{2, \ldots, 16\}$. A PC is fit to each component distribution $p_\phi(\mathbf{x}|\mathbf{z})$ and parameterised according to $\phi(\mathbf{z})$ where $\phi$ is a decoder with $\text{dom}(\phi) = \Omega_{\mathbf{Z}}$.

# Part 3: Continuous mixtures of PCs (motivation)

## Definition

A continuous mixture of probabilistic circuits (CMPC) is a PC of the form

$$p(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} p_\phi(\mathbf{x}|\mathbf{z}_i)$$

where $N \in \mathbb{N}$, $\mathbf{z}_1, \ldots, \mathbf{z}_N \sim p(\mathbf{z})$ with $\mathbf{Z} \sim \mathcal{N}(0, I_d)$ for some latent dimension $d \in \{2, \ldots, 16\}$. A PC is fit to each component distribution $p_\phi(\mathbf{x}|\mathbf{z})$ and parameterised according to $\phi(\mathbf{z})$ where $\phi$ is a decoder with $\mathrm{dom}(\phi) = \Omega_{\mathbf{Z}}$.

Answer: Choose a very simple sub-class of PC structures for each component distribution $p_\phi(\mathbf{x}|\mathbf{z})$, e.g. a fully-factorised model

$$p_\phi(\mathbf{x}|\mathbf{z}) = \prod_{i=1}^{n} p_\phi(x_i|\mathbf{z}).$$

# Part 3: Continuous mixtures of PCs (motivation)

## Definition

A continuous mixture of probabilistic circuits (CMPC) is a PC of the form

$$p(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} p_\phi(\mathbf{x}|\mathbf{z}_i)$$

where $N \in \mathbb{N}$, $\mathbf{z}_1, \ldots, \mathbf{z}_N \sim p(\mathbf{z})$ with $\mathbf{Z} \sim \mathcal{N}(0, I_d)$ for some latent dimension $d \in \{2, \ldots, 16\}$. A PC is fit to each component distribution $p_\phi(\mathbf{x}|\mathbf{z})$ and parameterised according to $\phi(\mathbf{z})$ where $\phi$ is a decoder with $\mathrm{dom}(\phi) = \Omega_{\mathbf{Z}}$.

## Terminology

CMPCs are a sub-class of discrete mixtures of PCs. Not 'continuous mixtures'...

# Part 3: Continuous mixtures of PCs (motivation)

More on the simple PC structures fit to the $N$ component distributions:

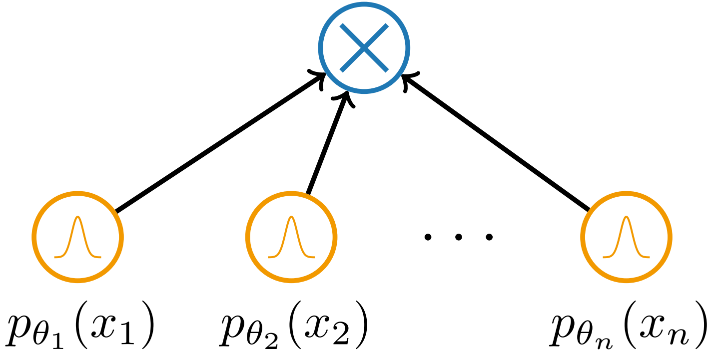



Figure: A fully-factorised model (FFM), $p(\mathbf{x}) = \prod_{i=1}^{n} p_{\theta_i}(x_i)$, as a PC.

# Part 3: Continuous mixtures of PCs (motivation)

More on the simple PC structures fit to the $N$ component distributions:

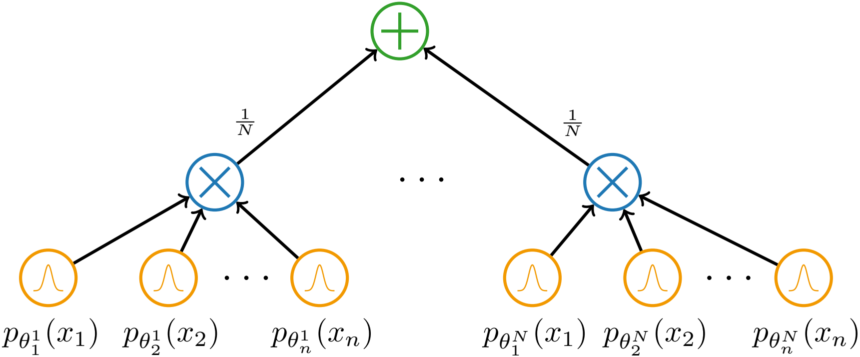



Figure: A CMPC with FFMs fit to the $N$ component distributions.

# Part 3: Continuous mixtures of PCs (motivation)

More on the simple PC structures fit to the $N$ component distributions:

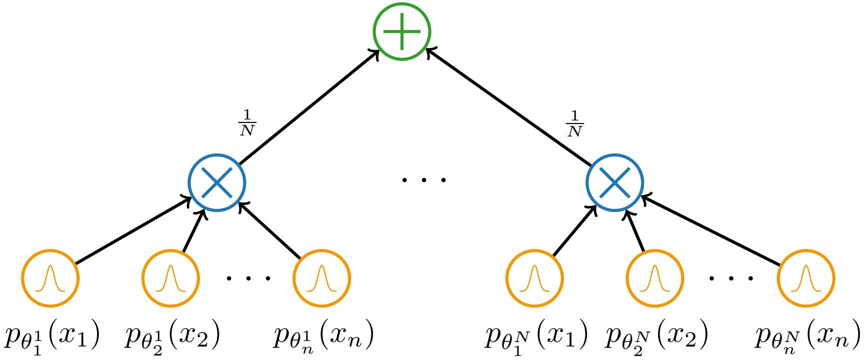



Figure: A CMPC with FFMs fit to the $N$ component distributions.

**No elaborate PC structure overall?**

More elaborate sub-classes of PCs could be used. Required compute increases.

# Part 3: Continuous mixtures of PCs (motivation)

Example (Binary MNIST):

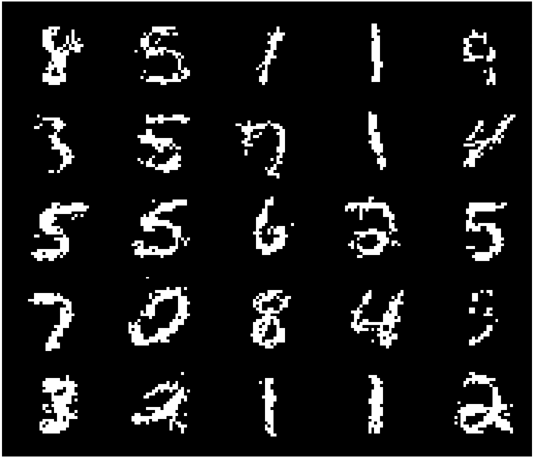


Figure: Samples of Binary MNIST. 28 by 28 (784 pixels) each.

# Part 3: Continuous mixtures of PCs (motivation)

Example (Binary MNIST): Fit an FFM to each component distribution. For component $i \in \{1, \ldots, N\}$, conditioned on sampled latent $\mathbf{z}_i \sim p(\mathbf{z})$, pixel distributions are independent Bernoulli distributions, i.e.

$$p_\phi(x_1, \ldots, x_{784}|\mathbf{z}_i) = \prod_{j=1}^{784} \mathsf{Ber}(x_j|\phi(\mathbf{z}_i))$$

and so a CMPC with $N = 100$ components is given by

$$\begin{aligned} p_\phi(x_1, \ldots, x_{784}) &= \frac{1}{100} \sum_{i=1}^{100} p_\phi(x_1, \ldots, x_{784}|\mathbf{z}_i) \\ &= \frac{1}{100} \sum_{i=1}^{100} \left[ \prod_{j=1}^{784} \mathsf{Ber}(x_j|\phi(\mathbf{z}_i)) \right]. \end{aligned}$$

If latent dimension $d = 16$ then $\mathbf{Z} \sim \mathcal{N}(0, I_{16})$ and $\phi : \mathbb{R}^{16} \to [0, 1]^{784}$.

# Part 3: Continuous mixtures of PCs (motivation)

Example (Binary MNIST): Fit an FFM to each component distribution. For component $i \in \{1, \ldots, N\}$, conditioned on sampled latent $\mathbf{z}_i \sim p(\mathbf{z})$, pixel distributions are independent Bernoulli distributions, i.e.

$$p_\phi(x_1, \ldots, x_{784} | \mathbf{z}_i) = \prod_{j=1}^{784} \mathsf{Ber}(x_j | \phi(\mathbf{z}_i))$$

and so a CMPC with $N = 100$ components is given by

$$\begin{aligned} p_\phi(x_1, \ldots, x_{784}) &= \frac{1}{100} \sum_{i=1}^{100} p_\phi(x_1, \ldots, x_{784} | \mathbf{z}_i) \\ &= \frac{1}{100} \sum_{i=1}^{100} \left[ \prod_{j=1}^{784} \mathsf{Ber}(x_j | \phi(\mathbf{z}_i)) \right]. \end{aligned}$$

If latent dimension $d = 16$ then $\mathbf{Z} \sim \mathcal{N}(0, I_{16})$ and $\phi : \mathbb{R}^{16} \to [0, 1]^{784}$.

# Part 3: Continuous mixtures of PCs (motivation)

Example (Binary MNIST): Fit an FFM to each component distribution. For component $i \in \{1, \ldots, N\}$, conditioned on sampled latent $\mathbf{z}_i \sim p(\mathbf{z})$, pixel distributions are independent Bernoulli distributions, i.e.

$$p_\phi(x_1, \ldots, x_{784} | \mathbf{z}_i) = \prod_{j=1}^{784} \mathsf{Ber}(x_j | \phi(\mathbf{z}_i))$$

and so a CMPC with $N = 100$ components is given by

$$\begin{aligned} p_\phi(x_1, \ldots, x_{784}) &= \frac{1}{100} \sum_{i=1}^{100} p_\phi(x_1, \ldots, x_{784} | \mathbf{z}_i) \\ &= \frac{1}{100} \sum_{i=1}^{100} \left[ \prod_{j=1}^{784} \mathsf{Ber}(x_j | \phi(\mathbf{z}_i)) \right]. \end{aligned}$$

If latent dimension $d = 16$ then $\mathbf{Z} \sim \mathcal{N}(0, I_{16})$ and $\phi : \mathbb{R}^{16} \to [0, 1]^{784}$.

# Part 3: Continuous mixtures of PCs (motivation)

## How might we fit the decoder $\phi$?

Task-dependent! For Binary MNIST, a multi-layer perceptron (MLP) works well. So an MLP encoding $\phi : \mathbb{R}^d \to [0, 1]^{784}$ where $d$ is the latent dimension.

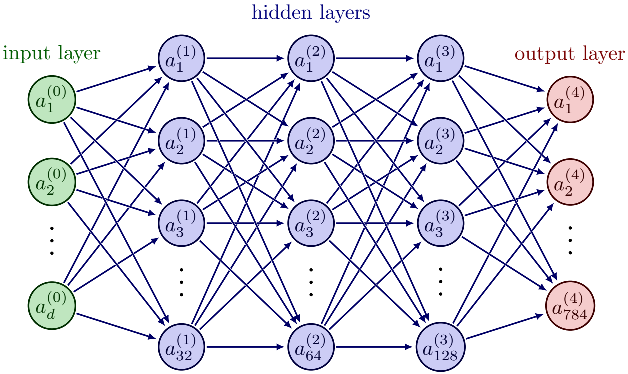



Figure: An MLP with $d$ input nodes and $784$ output nodes.

# Part 3: Continuous mixtures of PCs (motivation)

How to learn $\phi$? Maximum likelihood! Let $\{\mathbf{x}_j\}_{j=1}^{M}$ denote training data. Choose $N_{\text{train}}$ and latent dimension $d \in \{2, \ldots, , 16\}$ chosen a priori.

Sample $\mathbf{z}_1, \ldots \mathbf{z}_{N_{\text{train}}} \sim p(\mathbf{z})$, where $\mathbf{Z} \sim \mathcal{N}(0, I_d)$, and compute

$$\begin{aligned}
\arg\min_{\phi} \mathcal{L}(\phi) &= \arg\min_{\phi} \mathsf{NLL}(\phi) \\
&= \arg\min_{\phi} \left[ -\sum_{j=1}^{M} \log(p(\mathbf{x})) \right] \\
&= \arg\min_{\phi} \left[ -\sum_{j=1}^{M} \log \left( \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} p_{\phi}(\mathbf{x}_j | \mathbf{z}_i) \right) \right]
\end{aligned}$$

via gradient descent.

**Note:** If learned in batches, new $\mathbf{z}_1, \ldots, \mathbf{z}_{N_{\text{train}}} \sim p(\mathbf{z})$ for each batch.

# Part 3: Continuous mixtures of PCs (motivation)

How to learn $\phi$? Maximum likelihood! Let $\{\mathbf{x}_j\}_{j=1}^{M}$ denote training data. Choose $N_{\text{train}}$ and latent dimension $d \in \{2, \ldots, , 16\}$ chosen a priori.

Sample $\mathbf{z}_1, \ldots \mathbf{z}_{N_{\text{train}}} \sim p(\mathbf{z})$, where $\mathbf{Z} \sim \mathcal{N}(0, I_d)$, and compute

$$
\begin{aligned}
\arg\min_{\phi} \mathcal{L}(\phi) &= \arg\min_{\phi} \mathsf{NLL}(\phi) \\
&= \arg\min_{\phi} \left[ -\sum_{j=1}^{M} \log(p(\mathbf{x})) \right] \\
&= \arg\min_{\phi} \left[ -\sum_{j=1}^{M} \log \left( \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} p_{\phi}(\mathbf{x}_j | \mathbf{z}_i) \right) \right]
\end{aligned}
$$

via gradient descent.

**Note:** If learned in batches, new $\mathbf{z}_1, \ldots, \mathbf{z}_{N_{\text{train}}} \sim p(\mathbf{z})$ for each batch.

# Part 3: Continuous mixtures of PCs (motivation)

How to learn $\phi$? Maximum likelihood! Let $\{\mathbf{x}_j\}_{j=1}^{M}$ denote training data. Choose $N_{\text{train}}$ and latent dimension $d \in \{2, \ldots, , 16\}$ chosen a priori.

Sample $\mathbf{z}_1, \ldots \mathbf{z}_{N_{\text{train}}} \sim p(\mathbf{z})$, where $\mathbf{Z} \sim \mathcal{N}(0, I_d)$, and compute

$$\begin{aligned}
\arg\min_{\phi} \mathcal{L}(\phi) &= \arg\min_{\phi} \mathsf{NLL}(\phi) \\
&= \arg\min_{\phi} \left[ -\sum_{j=1}^{M} \log(p(\mathbf{x})) \right] \\
&= \arg\min_{\phi} \left[ -\sum_{j=1}^{M} \log \left( \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} p_{\phi}(\mathbf{x}_j | \mathbf{z}_i) \right) \right]
\end{aligned}$$

via gradient descent.

**Note:** If learned in batches, new $\mathbf{z}_1, \ldots, \mathbf{z}_{N_{\text{train}}} \sim p(\mathbf{z})$ for each batch.

# Part 3: Continuous mixtures of PCs (results)

After learning $\phi$: Choose $N_{\text{test}} \in \mathbb{N}$, sample $\mathbf{z}_1, \ldots, \mathbf{z}_{N_{\text{test}}} \sim p(\mathbf{z})$, compute $\phi(\mathbf{z}_1), \ldots, \phi(\mathbf{z}_{N_{\text{test}}})$ and compile

$$p(\mathbf{x}) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} p_\phi(\mathbf{x}|\mathbf{z}_i).$$

Note: Component distributions are parameterised by a shared decoder, so an $N-$component discrete mixture model is at least as expressive as an $N-$component CMPC. So what is useful about CMPCs?

# Part 3: Continuous mixtures of PCs (results)

After learning $\phi$: Choose $N_{\text{test}} \in \mathbb{N}$, sample $\mathbf{z}_1, \ldots, \mathbf{z}_{N_{\text{test}}} \sim p(\mathbf{z})$, compute $\phi(\mathbf{z}_1), \ldots, \phi(\mathbf{z}_{N_{\text{test}}})$ and compile

$$p(\mathbf{x}) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} p_\phi(\mathbf{x}|\mathbf{z}_i).$$

Note: Component distributions are parameterised by a shared decoder, so an $N-$component discrete mixture model is at least as expressive as an $N-$component CMPC. So what is useful about CMPCs?

# Part 3: Continuous mixtures of PCs (results)

Note: Component distributions are parameterised by a shared decoder, so an $N-$component discrete mixture model is at least as expressive as an $N-$component CMPC. So what is useful about CMPCs?

# Part 3: Continuous mixtures of PCs (results)

Note: Component distributions are parameterised by a shared decoder, so an $N-$component discrete mixture model is at least as expressive as an $N-$component CMPC. So what is useful about CMPCs?

| Dataset | BestPC | cm($S_{\mathrm{F}}$) | cm($S_{\mathrm{CLT}}$) | LO(cm($S_{\mathrm{CLT}}$)) | Dataset | BestPC | cm($S_{\mathrm{F}}$) | cm($S_{\mathrm{CLT}}$) | LO(cm($S_{\mathrm{CLT}}$)) |
|---|---|---|---|---|---|---|---|---|---|
| accid. | **26.74** | 33.27 | 28.69 | 28.81 | jester | 52.46 | **51.93** | 51.94 | 51.94 |
| ad | 16.07 | 18.71 | 14.76 | **14.42** | kdd | **2.12** | 2.13 | **2.12** | **2.12** |
| baudio | 39.77 | **39.02** | **39.02** | 39.04 | kosarek | 10.60 | 10.71 | 10.56 | **10.55** |
| bbc | 248.33 | **240.19** | 242.83 | 242.79 | msnbc | **6.03** | 6.14 | 6.05 | 6.05 |
| bnetflix | 56.27 | 55.49 | **55.31** | 55.36 | msweb | 9.73 | 9.68 | 9.62 | **9.60** |
| book | 33.83 | 33.67 | 33.75 | **33.55** | nltcs | **5.99** | **5.99** | **5.99** | **5.99** |
| c20ng | 151.47 | 148.24 | **148.17** | 148.28 | plants | 12.54 | 12.45 | **12.26** | 12.27 |
| cr52 | 83.35 | 81.52 | **81.17** | 81.31 | pumbs | **22.40** | 27.67 | 23.71 | 23.70 |
| cwebkb | 151.84 | 150.21 | 147.77 | **147.75** | tmovie | 50.81 | **48.69** | 49.23 | 49.29 |
| dna | **79.05** | 95.64 | 84.91 | 84.58 | tretail | 10.84 | 10.85 | 10.82 | **10.81** |

Table: Mean negative log-likelihoods attained by CMPCs on the test sets of 20 density estimation datasets. Taken from [2, Table 1].

# Part 4: Hybrid CMPCs (motivation)

CMPCs trained and benchmarked entirely generatively. No light shed on training and benchmarking them discriminatively. Discriminative PCs are of interest as they can classify incomplete samples (**marg** queries).

(1) How well can CMPCs classify samples of Binary MNIST?

Discriminative loss: replace NLL loss with cross-entropy, i.e. let $\{(\mathbf{x}_j, y_j)\}_{j=1}^{M}$ denote training data and compute

$$\begin{aligned}
\arg\min_{\phi} \mathcal{L}(\phi) &= \arg\min_{\phi} \mathrm{CE}(\phi) \\
&= \arg\min_{\phi} \left[ -\sum_{j=1}^{M} \log\left(p_{\phi}(y_j|\mathbf{x}_j)\right) \right] \\
&= \arg\min_{\phi} \left[ -\sum_{j=1}^{M} \log\left( \frac{\frac{1}{N}\sum_{i=1}^{N} p_{\phi}(\mathbf{x}_j, y_j|z_i)}{\frac{1}{N}\sum_{i=1}^{N} p_{\phi}(\mathbf{x}_j|z_i)} \right) \right].
\end{aligned}$$

# Part 4: Hybrid CMPCs (motivation)

CMPCs trained and benchmarked entirely generatively. No light shed on training and benchmarking them discriminatively. Discriminative PCs are of interest as they can classify incomplete samples (**marg** queries).

(1) How well can CMPCs classify samples of Binary MNIST?

Discriminative loss: replace NLL loss with cross-entropy, i.e. let $\{(\mathbf{x}_j, y_j)\}_{j=1}^{M}$ denote training data and compute

$$\begin{aligned}
\arg\min_{\phi} \mathcal{L}(\phi) &= \arg\min_{\phi} \mathrm{CE}(\phi) \\
&= \arg\min_{\phi} \left[ -\sum_{j=1}^{M} \log\left(p_\phi(y_j|\mathbf{x}_j)\right) \right] \\
&= \arg\min_{\phi} \left[ -\sum_{j=1}^{M} \log\left( \frac{\frac{1}{N}\sum_{i=1}^{N} p_\phi(\mathbf{x}_j, y_j|z_i)}{\frac{1}{N}\sum_{i=1}^{N} p_\phi(\mathbf{x}_j|z_i)} \right) \right].
\end{aligned}$$

# Part 4: Hybrid CMPCs (motivation)

CMPCs trained and benchmarked entirely generatively. No light shed on training and benchmarking them discriminatively. Discriminative PCs are of interest as they can classify incomplete samples (**marg** queries).

(1) How well can CMPCs classify samples of Binary MNIST?

Discriminative loss: replace NLL loss with cross-entropy, i.e. let $\{(\mathbf{x}_j, y_j)\}_{j=1}^{M}$ denote training data and compute

$$\begin{aligned}
\arg\min_{\phi} \mathcal{L}(\phi) &= \arg\min_{\phi} \mathsf{CE}(\phi) \\
&= \arg\min_{\phi} \left[ -\sum_{j=1}^{M} \log\left(p_{\phi}(y_j|\mathbf{x}_j)\right) \right] \\
&= \arg\min_{\phi} \left[ -\sum_{j=1}^{M} \log\left( \frac{\frac{1}{N}\sum_{i=1}^{N} p_{\phi}(\mathbf{x}_j, y_j|z_i)}{\frac{1}{N}\sum_{i=1}^{N} p_{\phi}(\mathbf{x}_j|z_i)} \right) \right].
\end{aligned}$$

# Part 4: Hybrid CMPCs (motivation)

(2) Can generative power be maintained if learned discriminatively?

Hybrid loss [1]:

$$\mathcal{L}_\lambda(\phi) = \lambda \cdot \mathsf{CE}(\phi) + (1-\lambda) \cdot \mathsf{NLL}(\phi)$$

where $\lambda \in [0,1]$ determines the extent to which we train discriminatively (cross-entropy) and generatively (negative log-likelihood).

We call these hybrid CMPCs:

- $\lambda = 0 \quad \rightarrow$ hybrid CMPC trained fully-generative
- $\lambda = 1 \quad \rightarrow$ hybrid CMPC trained fully-discriminative
- $\lambda = 0.5 \rightarrow$ hybrid CMPC trained with a mix of gen. and discrim.

Question: Is there some $\lambda \in [0,1]$ such that the model does well generatively and discriminatively?

# Part 4: Hybrid CMPCs (motivation)

(2) Can generative power be maintained if learned discriminatively?

Hybrid loss [1]:

$$\mathcal{L}_\lambda(\phi) = \lambda \cdot \mathsf{CE}(\phi) + (1-\lambda) \cdot \mathsf{NLL}(\phi)$$

where $\lambda \in [0,1]$ determines the extent to which we train discriminatively (cross-entropy) and generatively (negative log-likelihood).

We call these hybrid CMPCs:

- $\lambda = 0 \quad \rightarrow$ hybrid CMPC trained fully-generative
- $\lambda = 1 \quad \rightarrow$ hybrid CMPC trained fully-discriminative
- $\lambda = 0.5 \rightarrow$ hybrid CMPC trained with a mix of gen. and discrim.

Question: Is there some $\lambda \in [0,1]$ such that the model does well generatively and discriminatively?

# Part 4: Hybrid CMPCs (motivation)

(2) Can generative power be maintained if learned discriminatively?

Hybrid loss [1]:

$$\mathcal{L}_\lambda(\phi) = \lambda \cdot \mathsf{CE}(\phi) + (1-\lambda) \cdot \mathsf{NLL}(\phi)$$

where $\lambda \in [0,1]$ determines the extent to which we train discriminatively (cross-entropy) and generatively (negative log-likelihood).

We call these hybrid CMPCs:

- $\lambda = 0 \quad \rightarrow$ hybrid CMPC trained fully-generative
- $\lambda = 1 \quad \rightarrow$ hybrid CMPC trained fully-discriminative
- $\lambda = 0.5 \rightarrow$ hybrid CMPC trained with a mix of gen. and discrim.

Question: Is there some $\lambda \in [0,1]$ such that the model does well generatively and discriminatively?

# Part 4: Hybrid CMPCs (results)

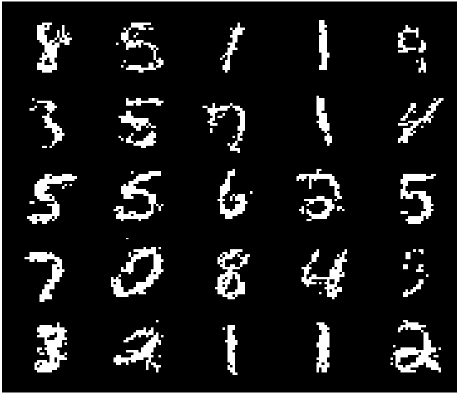


Figure: Samples of Binary MNIST.

Some details:

- 50,000 training samples, 10,000 validation and 10,000 testing
- 28 by 28 each, pixel values in $\{0, 1\}$, i.e. $\Omega_{\mathbf{X}} = \{0, 1\}^{784}$
- 10 classes, i.e. $\Omega_Y = \{0, \ldots, 9\}$

# Part 4: Hybrid CMPCs (results)

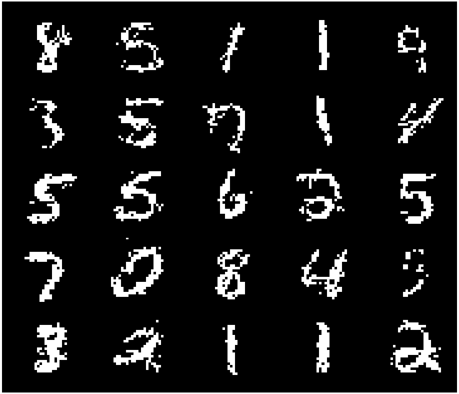


Figure: Samples of Binary MNIST.

Some details:

- 50,000 training samples, 10,000 validation and 10,000 testing
- 28 by 28 each, pixel values in $\{0, 1\}$, i.e. $\Omega_{\mathbf{X}} = \{0, 1\}^{784}$
- 10 classes, i.e. $\Omega_Y = \{0, \ldots, 9\}$

# Part 4: Hybrid CMPCs (classification results)

Six hybrid CMPCs trained, one for each $\lambda \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$, with:

- Latent dimension $d = 16$, so $\mathbf{Z} \sim \mathcal{N}(0, I_{16})$ and $\phi : \mathbb{R}^{16} \to [0,1]^{784}$
- $N_{\text{train}} = 2^{13}$ components during training

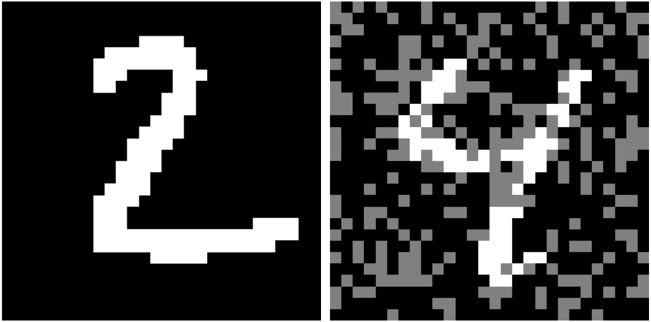


Figure: Samples of Binary MNIST. Left: A regular sample. Right: A sample pertaining to the digit 4 in which 30% of pixel values are missing at random.

# Part 4: Hybrid CMPCs (classification results)

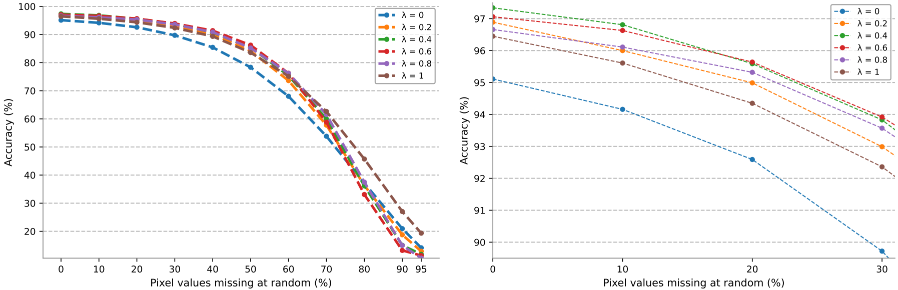



Figure: Classification accuracies of hybrid CMPCs trained on Binary MNIST for $\lambda \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$ with differing portions of missing pixel values.

## Surprising result

The hybrid CMPC trained with $\lambda = 0.4$ performs the best during classification, not the one trained entirely discriminatively (with $\lambda = 1$).

# Part 4: Hybrid CMPCs (classification results)

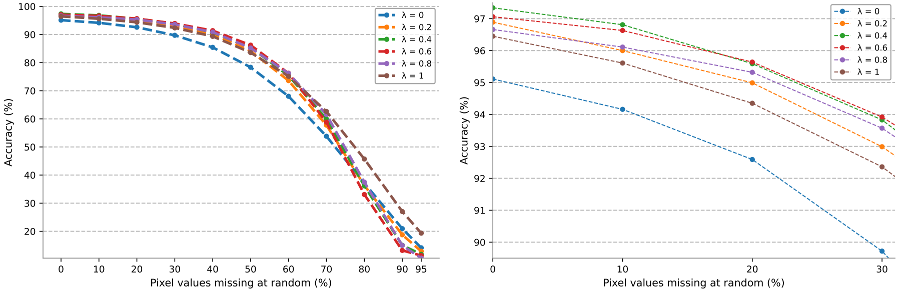



Figure: Classification accuracies of hybrid CMPCs trained on Binary MNIST for $\lambda \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$ with differing portions of missing pixel values.

## Surprising result

The hybrid CMPC trained with $\lambda = 0.4$ performs the best during classification, not the one trained entirely discriminatively (with $\lambda = 1$).

# Part 4: Hybrid CMPCs (sampling results)

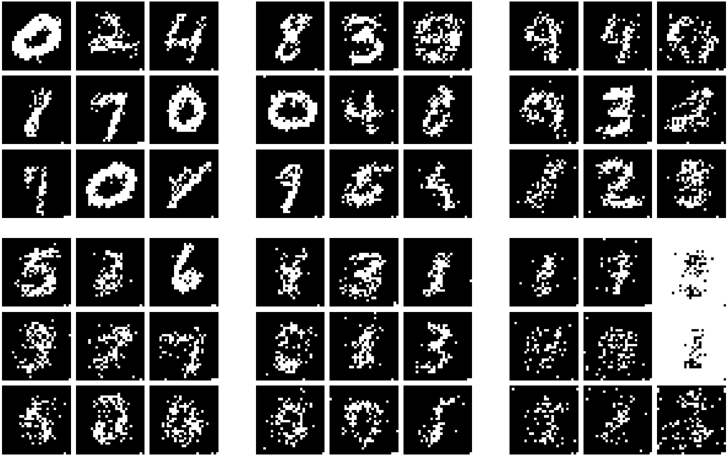


Figure: Samples of Binary MNIST drawn from hybrid CMPCs with $d = 16$ and $\lambda \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$ ordered left-to-right top-to-bottom.

# Discussion & Conclusion

Contributions:

1. Beginner-friendly overviews of PCs and CMPCs, of which there are currently none
2. Experimentation regarding learning CMPCs in a hybrid manner

# Discussion & Conclusion

Contributions:

1. Beginner-friendly overviews of PCs and CMPCs, of which there are currently none
2. Experimentation regarding learning CMPCs in a hybrid manner

Main results:

1. Highest classification accuracies given by $\lambda = 0.4$; higher than $\lambda = 1$
2. Sample degradation is gradual, heavy pixelation from $\lambda = 0.6$ upward. Trading generative ability for discriminative via $\lambda$ is possible

# Discussion & Conclusion

Contributions:

1. Beginner-friendly overviews of PCs and CMPCs, of which there are currently none
2. Experimentation regarding learning CMPCs in a hybrid manner

Main results:

1. Highest classification accuracies given by $\lambda = 0.4$; higher than $\lambda = 1$
2. Sample degradation is gradual, heavy pixelation from $\lambda = 0.6$ upward. Trading generative ability for discriminative via $\lambda$ is possible

Future work:

1. Fit more sophisticated PC structures fit to the $N$ components
2. Fit CMPCs to more sophisticated datasets, e.g. CIFAR-10

# References

[1] G. Bouchard and B. Triggs. The tradeoff between generative and discriminative classifiers. In *16th IASC International Symposium on Computational Statistics*, pages 721–728, 2004.

[2] A. H. Correia, G. Gala, E. Quaeghebeur, C. de Campos, and R. Peharz. Continuous mixtures of tractable probabilistic models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7244–7252, 2023.

[3] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

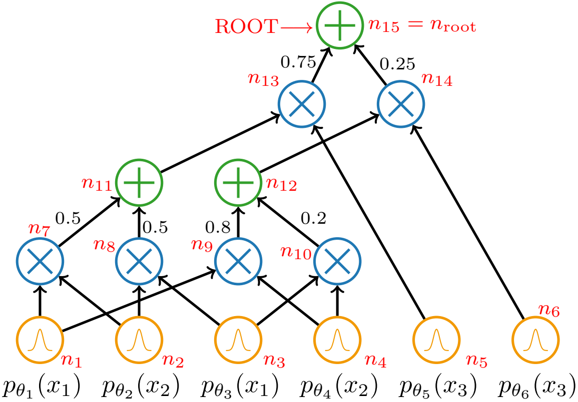
[4] R. Peharz, R. Gens, F. Pernkopf, and P. Domingos. On the latent variable interpretation in sum-product networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(10):2030–2044, 2016.

[5] D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996.

# Appendix: The scope of a node

## Scope

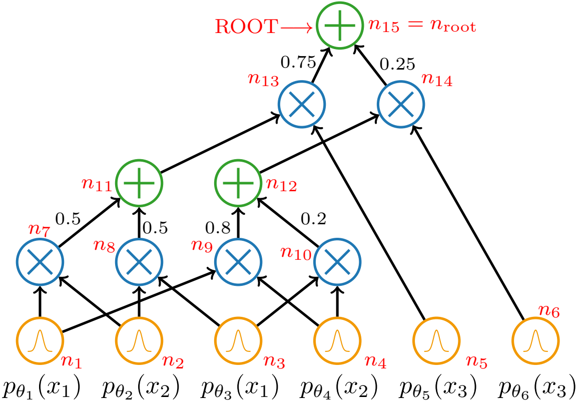
Recursively, the scope $\mathrm{sc}(n_i)$ of a sum or product node $n_i$ is the union of the scopes of its parents. The scope of a input node is its corresponding subset of random variables in $\mathbf{X}$. By construction, $\mathrm{sc}(n_{\text{root}}) = \mathbf{X}$.



# Appendix: The scope of a node

## Scope

Recursively, the scope $\mathsf{sc}(n_i)$ of a sum or product node $n_i$ is the union of the scopes of its parents. The scope of a input node is its corresponding subset of random variables in $\mathbf{X}$. By construction, $\mathsf{sc}(n_{\mathsf{root}}) = \mathbf{X}$.

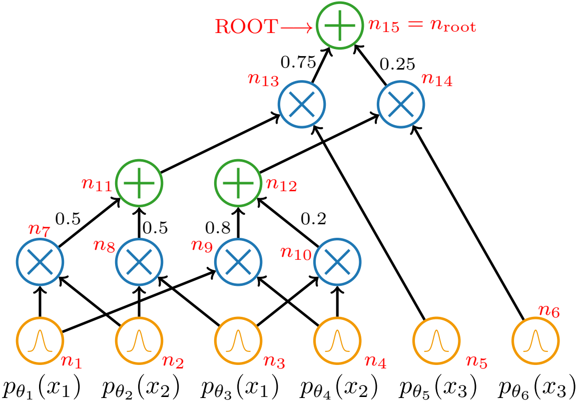



$\mathsf{sc}(n_3) = \{X_1\}$

# Appendix: The scope of a node

## Scope

Recursively, the scope $\mathsf{sc}(n_i)$ of a sum or product node $n_i$ is the union of the scopes of its parents. The scope of a input node is its corresponding subset of random variables in $\mathbf{X}$. By construction, $\mathsf{sc}(n_{\mathsf{root}}) = \mathbf{X}$.



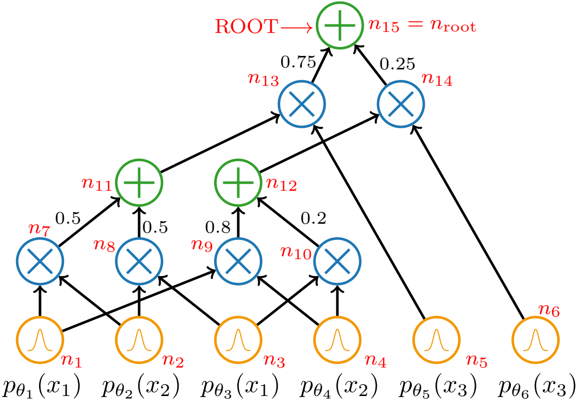

$$\mathsf{sc}(n_3) = \{X_1\}$$

$$\begin{aligned}\mathsf{sc}(n_8) &= \mathsf{sc}(n_2) \cup \mathsf{sc}(n_3) \\ &= \{X_2\} \cup \{X_1\} \\ &= \{X_1, X_2\}\end{aligned}$$

# Appendix: The scope of a node

## Scope

Recursively, the scope $\mathrm{sc}(n_i)$ of a sum or product node $n_i$ is the union of the scopes of its parents. The scope of a input node is its corresponding subset of random variables in $\mathbf{X}$. By construction, $\mathrm{sc}(n_{\mathrm{root}}) = \mathbf{X}$.




$$\mathrm{sc}(n_3) = \{X_1\}$$

$$\begin{aligned}\mathrm{sc}(n_8) &= \mathrm{sc}(n_2) \cup \mathrm{sc}(n_3) \\ &= \{X_2\} \cup \{X_1\} \\ &= \{X_1, X_2\}\end{aligned}$$

$$\begin{aligned}\mathrm{sc}(n_{15}) &= \mathrm{sc}(n_{\mathrm{root}}) \\ &= \{X_1, X_2, X_3\} \\ &= \mathbf{X}\end{aligned}$$

# Appendix: Smoothness

## Smoothness

A PC is smooth if for any sum node in $\mathcal{G}$, the scopes of its parents are equal.

## Smoothness

A PC is smooth if for any sum node in $\mathcal{G}$, the scopes of its parents are equal.

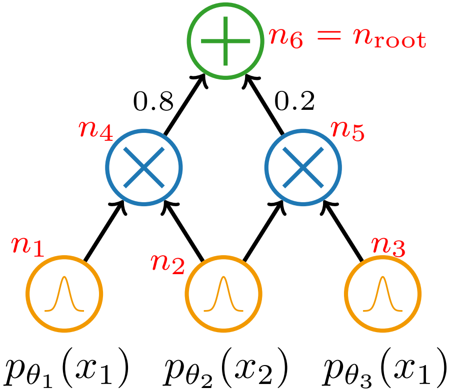



Figure: A smooth PC.

Example: $\textbf{pa}(n_6) = \{n_4, n_5\}$, $\text{sc}(n_4) = \{X_1, X_2\}$ and $\text{sc}(n_5) = \{X_1, X_2\}$

# Appendix: Decomposability

## Decomposability

A PC is decomposable if for any product node in $\mathcal{G}$, the scopes of its parents are pairwise disjoint.

# Appendix: Decomposability

## Decomposability

A PC is decomposable if for any product node in $\mathcal{G}$, the scopes of its parents are pairwise disjoint.

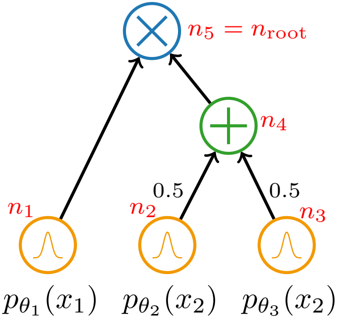



Figure: A decomposable PC.

Example: $\textbf{pa}(n_5) = \{n_1, n_4\}$, $\text{sc}(n_1) = \{X_1\}$ and $\text{sc}(n_4) = \{X_2\}$

# Appendix: Decomposability

## Decomposability

A PC is decomposable if for any product node in $\mathcal{G}$, the scopes of its parents are pairwise disjoint.

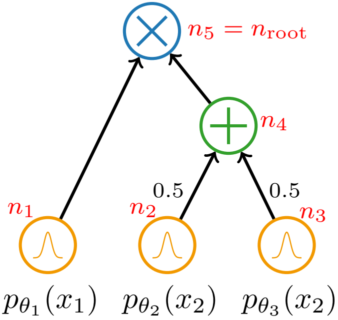



Figure: A decomposable PC.

The associated function $p_{n_{\mathrm{root}}} : \Omega_{\mathbf{X}} \to \mathbb{R}_{\geq 0}$ of the root node of a smooth and decomposable PC $(\mathbf{X}, p_{(\mathcal{G},\mathbf{w},\theta)})$ is a probability function over $\mathbf{X}$.

# Appendix: Pushing down integrals

## Purpose of smoothness and decomposability

The associated function $p_{n_{\text{root}}} : \Omega_{\mathbf{X}} \to \mathbb{R}_{\geq 0}$ of the root node of a smooth and decomposable PC $(\mathbf{X}, p_{(\mathcal{G},\mathbf{w},\theta)})$ is a probability function over $\mathbf{X}$.

For $X_k \in$ sc(sum node):

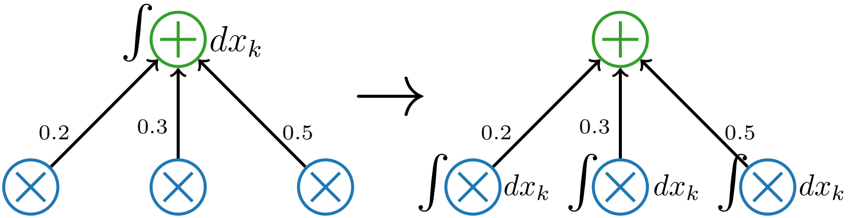



Figure: An integral being pushed down from a sum node to its parents.

$$\int_{\Omega_{X_2}} \sum_{i=1}^{3} p_i(x_1, x_2, x_3)\,dx_2 \longrightarrow \sum_{i=1}^{3} \int_{\Omega_{X_2}} p_i(x_1, x_2, x_3)\,dx_2$$

# Appendix: Pushing down integrals

## Purpose of smoothness and decomposability

The associated function $p_{n_{\text{root}}} : \Omega_{\mathbf{X}} \to \mathbb{R}_{\geq 0}$ of the root node of a smooth and decomposable PC $(\mathbf{X}, p_{(\mathcal{G},\mathbf{w},\theta)})$ is a probability function over $\mathbf{X}$.

For $X_k \in$ sc(sum node):

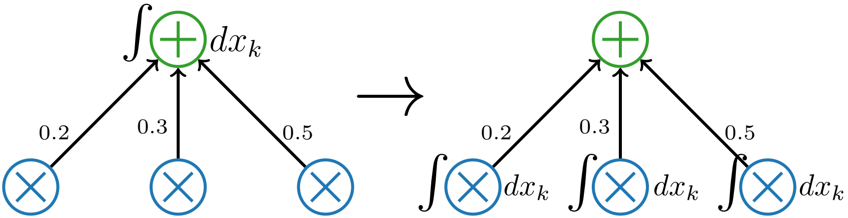



Figure: An integral being pushed down from a sum node to its parents.

$$\int_{\Omega_{X_2}} \sum_{i=1}^{3} p_i(x_1, x_2, x_3) dx_2 \longrightarrow \sum_{i=1}^{3} \int_{\Omega_{X_2}} p_i(x_1, x_2, x_3) dx_2$$

# Appendix: Pushing down integrals

## Purpose of smoothness and decomposability

The associated function $p_{n_{\text{root}}} : \Omega_{\mathbf{X}} \to \mathbb{R}_{\geq 0}$ of the root node of a smooth and decomposable PC $(\mathbf{X}, p_{(\mathcal{G},\mathbf{w},\theta)})$ is a probability function over $\mathbf{X}$.

For $X_k \in$ sc(sum node):

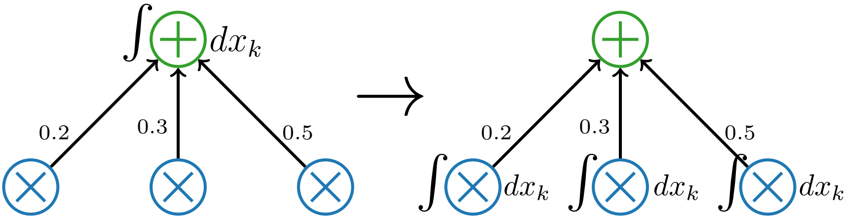



Figure: An integral being pushed down from a sum node to its parents.

$$\int_{\Omega_{X_2}} \sum_{i=1}^{3} p_i(x_1, x_2, x_3) dx_2 \longrightarrow \sum_{i=1}^{3} \int_{\Omega_{X_2}} p_i(x_1, x_2, x_3) dx_2$$

# Appendix: Pushing down integrals

## Purpose of smoothness and decomposability

The associated function $p_{n_{\text{root}}} : \Omega_{\mathbf{X}} \to \mathbb{R}_{\geq 0}$ of the root node of a smooth and decomposable PC $(\mathbf{X}, p_{(\mathcal{G},\mathbf{w},\theta)})$ is a probability function over $\mathbf{X}$.

For $X_k \in$ sc(product node):

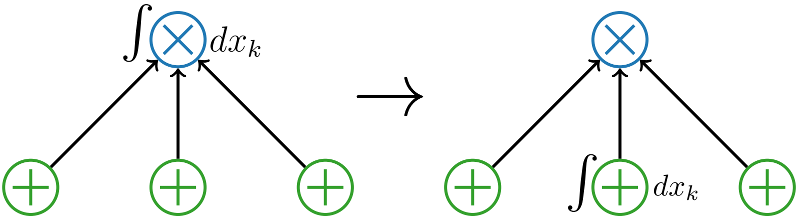


Figure: An integral being pushed down from a product node to one of its parents.

$$\int_{\Omega_{X_2}} p_1(x_1, x_2) p_2(x_3, x_4) p_3(x_5) dx_2 \to p_2(x_3, x_4) p_3(x_5) \int_{\Omega_{X_2}} p_1(x_1, x_2) dx_2$$

# Appendix: Pushing down integrals

## Purpose of smoothness and decomposability

The associated function $p_{n_{\text{root}}} : \Omega_{\mathbf{X}} \to \mathbb{R}_{\geq 0}$ of the root node of a smooth and decomposable PC $(\mathbf{X}, p_{(\mathcal{G},\mathbf{w},\theta)})$ is a probability function over $\mathbf{X}$.

For $X_k \in$ sc(product node):

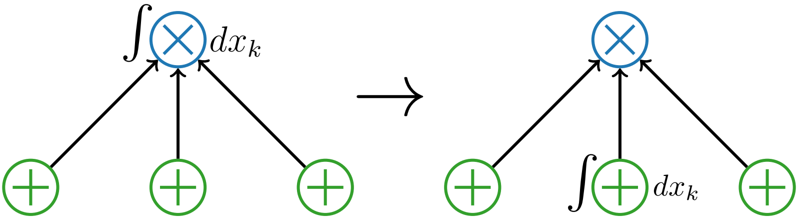



Figure: An integral being pushed down from a product node to one of its parents.

$$\int_{\Omega_{X_2}} p_1(x_1, x_2) p_2(x_3, x_4) p_3(x_5) dx_2 \to p_2(x_3, x_4) p_3(x_5) \int_{\Omega_{X_2}} p_1(x_1, x_2) dx_2$$

# Appendix: Pushing down integrals

## Purpose of smoothness and decomposability

The associated function $p_{n_{\text{root}}} : \Omega_{\mathbf{X}} \to \mathbb{R}_{\geq 0}$ of the root node of a smooth and decomposable PC $(\mathbf{X}, p_{(\mathcal{G},\mathbf{w},\theta)})$ is a probability function over $\mathbf{X}$.

For $X_k \in$ sc(product node):

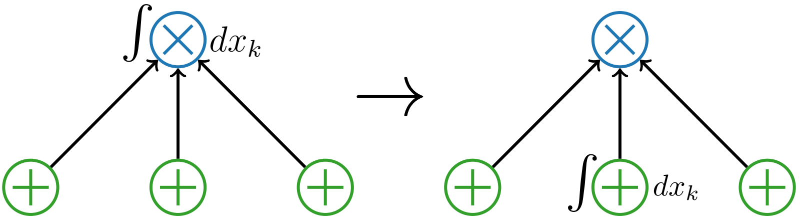



Figure: An integral being pushed down from a product node to one of its parents.

$$\int_{\Omega_{X_2}} p_1(x_1, x_2) p_2(x_3, x_4) p_3(x_5) dx_2 \rightarrow p_2(x_3, x_4) p_3(x_5) \int_{\Omega_{X_2}} p_1(x_1, x_2) dx_2$$

# Appendix: Pushing down integrals

## Purpose of smoothness and decomposability

The associated function $p_{n_{\text{root}}} : \Omega_{\mathbf{X}} \to \mathbb{R}_{\geq 0}$ of the root node of a smooth and decomposable PC $(\mathbf{X}, p_{(\mathcal{G},\mathbf{w},\theta)})$ is a probability function over $\mathbf{X}$.

For $X_k \in$ sc(input node):

$\int$  $dx_k = 1$

Figure: An integral being evaluated over the associated function of an input node.

$$\int_{\Omega_{X_2}} p_{\theta_z}(x_2)dx_2 \longrightarrow 1$$

# Appendix: Pushing down integrals

## Purpose of smoothness and decomposability

The associated function $p_{n_{\text{root}}} : \Omega_{\mathbf{X}} \to \mathbb{R}_{\geq 0}$ of the root node of a smooth and decomposable PC $(\mathbf{X}, p_{(\mathcal{G},\mathbf{w},\theta)})$ is a probability function over $\mathbf{X}$.

For $X_k \in$ sc(input node):

Figure: An integral being evaluated over the associated function of an input node.

$$\int_{\Omega_{X_2}} p_{\theta_z}(x_2) dx_2 \longrightarrow 1$$

# Appendix: Pushing down integrals

## Purpose of smoothness and decomposability

The associated function $p_{n_{\text{root}}} : \Omega_{\mathbf{X}} \to \mathbb{R}_{\geq 0}$ of the root node of a smooth and decomposable PC $(\mathbf{X}, p_{(\mathcal{G}, \mathbf{w}, \theta)})$ is a probability function over $\mathbf{X}$.

For $X_k \in$ sc(input node):

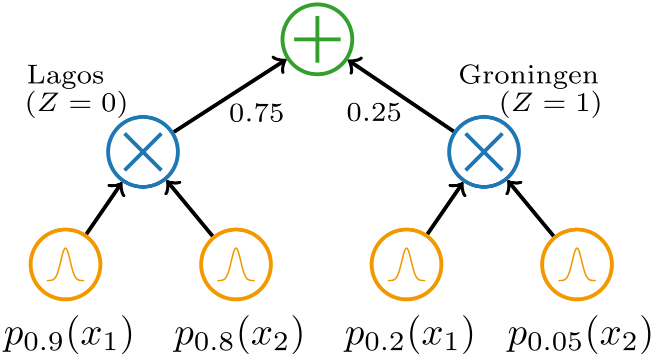
$$\int$$

$$dx_k = 1$$

Figure: An integral being evaluated over the associated function of an input node.

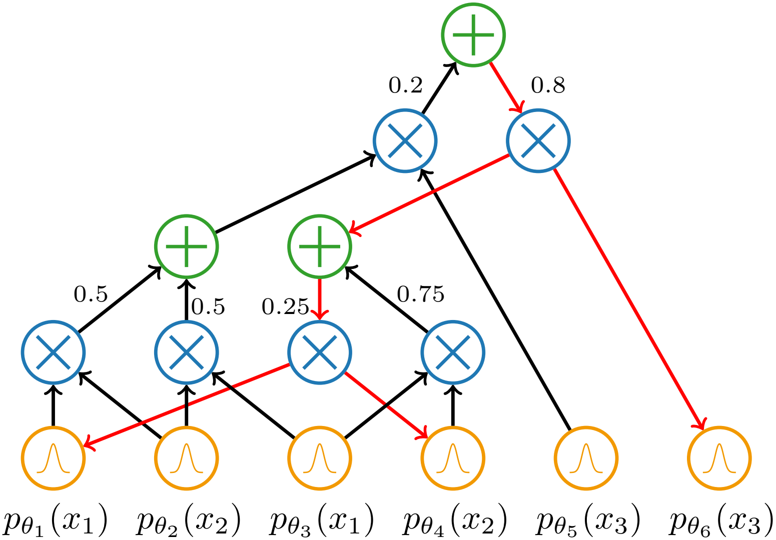
$$\int_{\Omega_{X_2}} p_{\theta_z}(x_2) dx_2 \longrightarrow 1$$

# Appendix: LVM interpretation of PCs [4]

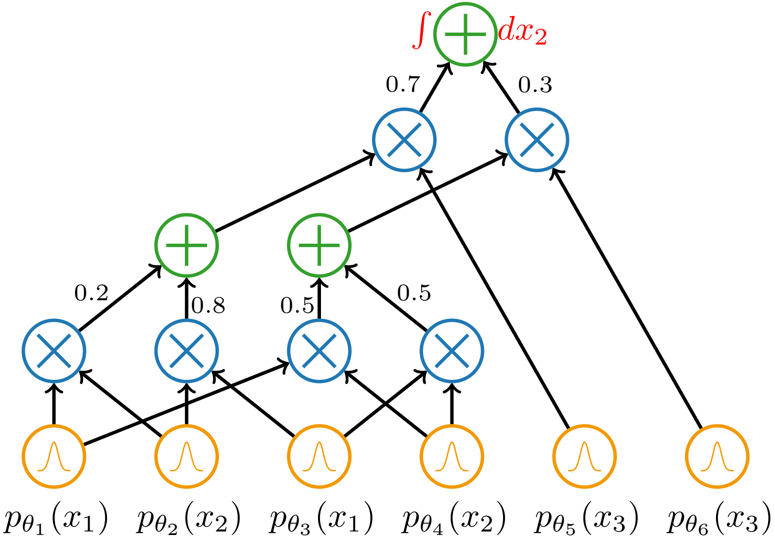



The latent variable $\mathbf{Z} \sim \text{Ber}(0.25)$ pertains to location of data recording and is marginalised out.

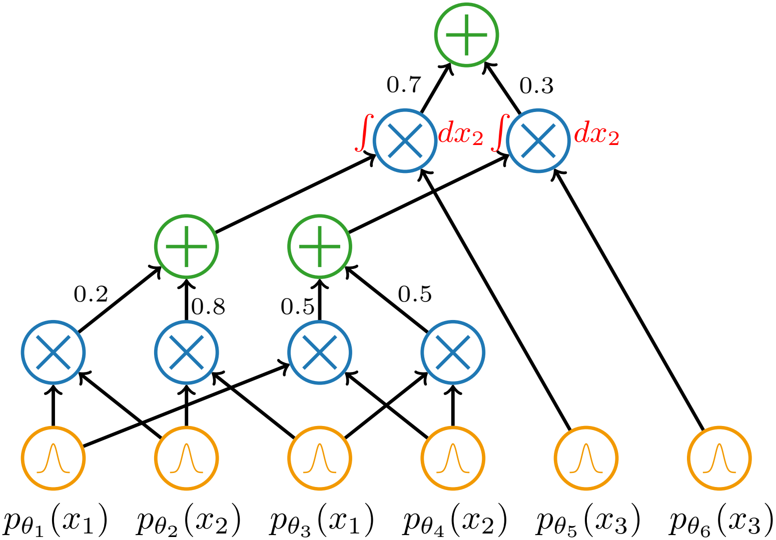
# Appendix: Sampling from a PC



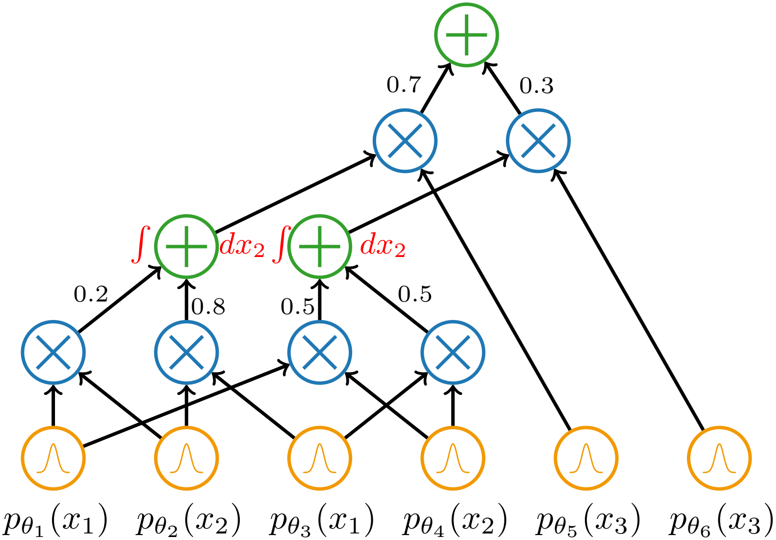
# Appendix: **marg** queries



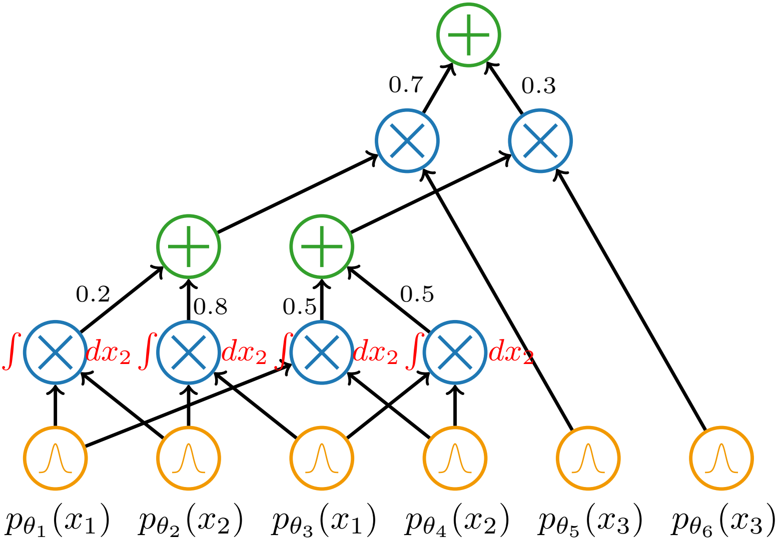
# Appendix: **marg** queries



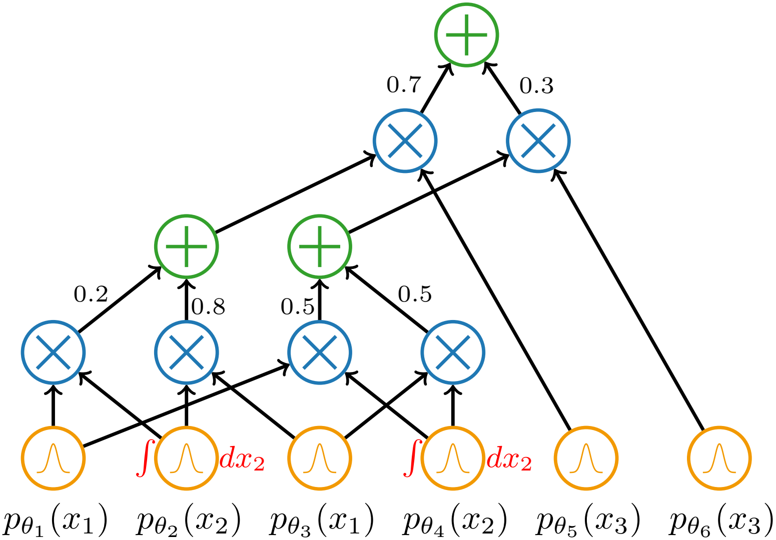
# Appendix: **marg** queries



# Appendix: **marg** queries



# Appendix: **marg** queries



# Appendix: **marg** queries